HIGH PRECISION TREATMENT FOR FRACTIONATED STEREOTACTIC RADIOTHERAPY

By

CHING-CHONG YANG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1994

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to the members of my supervisory committee for their kind help and guidance throughout this work. I would also like to express special thanks to my committee chairman, Dr. Francis Bova, for his invaluable suggestions and sincere instruction through this research. I would also like to give special thanks to Dr. Carl Crane for invaluable initiation about the solving matrix problem. Also I would like to thank Dr. William Friedman for his kind support for the whole project and Dr. William Mendenhall for his comments on the radiobiological aspectes of this project. Special thanks should be given to Dr. Lawrence Fitzgerald for his constant encouragement and comments throughout the research period. I also would like to thank Professor James Tulenko and Dr. William Properzio for discussions, corrections, and encouragement.

I would like to give special thanks to Ms. Pam Mydock for her excellent job in proofreading the draft of this manuscript. I also could not forget my fellow graduate students and the physicists, engineers, and doctors at the Shands Cancer Center who encouraged me to go on for my graduate career: Kevin Kalbaugh, Linda Ewald, Rene Craig, Ruth Ann Good, Fred Harmon, Sanford Meeks, Soon Huh, Dale Moss, Don Dubois, Revlon Williams, Duane Sandene and Drs. Nancy Mendenhall, Jim Parsons, Michael Sombeck, Jatinder Palta, and Chihray Liu. Finally I would like to thank my sweet Ph.D. wife to has accompanied me through the years without any complaint and my parents for their full support, both spiritually and financially.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

HIGH PRECISION TREATMENT FOR
FRACTIONATED STEREOTACTIC RADIOTHERAPY

By

Ching-Chong Yang

April, 1994

Chairperson: Frank J. Bova
Major Department: Nuclear Engineering Sciences

Fractionated Stereotactic Radiotherapy (FSR) refers to high precision, multifraction, external-beam irradiation that utilizes accurate refixation techniques. Three optimization algorithms were developed to calculate six-degrees-of-freedom motion parameters for patient repositioning in FSR. These algorithms provide the parameters for patient rotation and translation. Simulation results have shown a maximum target displacement error of 0.4 mm and an orientation error of 0.01 degree. Phantom image and real-time 3-D data studies have been performed to verify the accuracy of the process. A new repositioning system is proposed that provides for high rotational and translational precision in fractionated treatment. An independent verification procedure is also developed and verified to assure the high treatment precision in the intracranial region.

CHAPTER 1
INTRODUCTION

High Precision Radiation Therapy

Historically, a large single dose divided into many small fractions has been employed in cancer treatment. This is to maximize the tumor dose and to take advantage of the differential radiobiological responses between tumor and normal tissues. In order to achieve the constant treatment accuracy for each treatment course, various fixation methods have been developed to maintain reproducible patient treatment position. Depending upon the different treatment area, treatment errors from several millimeters up to a centimeter have been observed [Rab85]. In order to achieve high precision in radiation therapy positioning, a treatment system with high target localization and irradiation accuracies has been developed. The newly developed irradiation technique is termed stereotactic radiosurgery since it performs noncoplanar radiation treatment without invasive open surgery for intracranial diseases. With the use of a reference frame attached to the patient's skull, stereotactic radiosurgery allows high precision for both target localization and treatment. The fixation technique of radiosurgery is invasive and it is thus not easy to perform dose fractionation. Therefore, a large, single fraction of exposure is currently administered in a stereotactic radiosurgery procedure, thus the radiobiological advantages of fractionation have not yet been realized.

Stereotactic radiosurgery is used to treat patients who are not good candidates for conventional neurosurgery. A single high dose delivery to a vascular lesion has been acknowledged as an effective method to cause lesion thromboses. Through the invasive fixation of a halo-shaped of metal immobilization device, the patient's anatomy relative to the Brown-Roberts-Wells® (BRW) neurosurgical stereotactic reference ring and frame system is fixed, and a CT image set with multiple slices is performed without removing the head ring. The images provide geometrical relationships of the head relative to the head ring and thus any point in the head can be assigned an X, Y, Z coordinate relative to the known geometry on the ring. Once the target coordinate related to the head ring is known, the treatment planning process can begin. After interactive optimization of treatment planning, the dose prescription is selected and patient treatment proceeds. Radiation is delivered to a point whose reference coordinates are preserved through the planning and treatment processes since the head ring remains fixed on the patient until treatment is complete.

With a single fraction stereotactic treatment, the prescribed dose can be delivered exactly to the target area. However, once the ring is removed from the patient, the relative coordinates disappear and then fractionation becomes impossible. The second fraction of radiation could be provided by repeating the same invasive localization procedure. Clinically, this is impractical. Therefore, stereotactic radiosurgery achieves high precision but sacrifices the radiobiological differentiation between tumor and normal tissues. Stereotactic radiosurgery manipulates the treatment planning design by direct implementation of dosimetric information on the image basis. This virtual simulation procedure involves no patient contact while obtaining the best non-coplnar radiation beam combinations on the computer. Virtual simulation differs from most of the radiation oncology routine practices by replacing conventional treatment simulation with 3-D image data basis and computer software. The same geometry used in imaging the patient will be used in treatment, and the data sets provide the basic information for beam delivery. Application of virtual simulation requires the capabilities of transferring the treatment planning geometry from the computer to the treatment position in a way which is accurate, reproducible, and efficient [She90]. The mathematical projection of radiation beams to the target can be transferred to the exact position under the treatment machine since the mathematical simulation process on computer conveys the same information in the real treatment condition.

Radiobiology for Fractionated Treatment

The single fraction treatment for radiosurgery of Arterial Venous Malformation (AVM) has been proved to be clinically very effective [Fri92a]. The AVM suffers sufficient cellular damage from a single dose of radiation to the vascular structure that over a period of many months the abnormal vessels thrombose. However, the use of a single large dose for the treatment of malignancies is contrary to all the radiotherapy experience that has been accumulated over the past decade. In the treatment of a malignancy, the therapeutic ratio can be greatly improved by fractionation. Fractionation increases the cellular depopulation of the tumor because of the reoxygenation while reducing the damage to critical late responding normal tissues. The biological principle of fractionated treatment of tumors is based on the experimental evidence that there is a difference in shape between the dose-response characteristics of early responding tissues and tumors and late responding tissues [Hal93]. Therefore, treating malignant tumors with a single fraction will result in a suboptimal therapeutic gain between local tumor control and the normal tissue late effects even for small tumors. An improved therapeutic ratio is expected if the treatments are fractionated.

Patient Repositioning

Radiation therapy in cancer management is a complex process that involves many procedures to achieve the best treatment results. However, during the process, inaccuracies will occur due to a variety of procedures, such as localization of tumor volume, treatment setups, and immobilization and repositioning techniques.

Routine radiation therapy is usually administered under multifractionated treatments. During the treatment process, special attention is required in four different areas. First, the prescribed doses should be defined based upon clinical research and continuous evaluation; this is beyond the scope of our discussion. Second, the machine output and dose calculation should be correctly specified. In this area, errors are already minimized due to the improvement of dosimetry systems, NIST[1] traceable ionization chambers, an improved dosimetry protocol such as TG-21 [Tas83], and redundant calibration procedures by medical physicists. The third and the fourth requirements involve the technical setup of the daily treatment and accurate dose delivery to the tumor for the duration of each treatment. Among these steps, tumor localization and radiation field misalignment are the most error-prone factors for radiation therapy. Errors in

[1]NIST: National Institute of Standards and Technology, Gaithersburg, MD.

localization can lead to reduced doses at the sites of tumors or to excessive doses to normal tissues. If the target is missed, doses will be delivered to normal tissues, which increases the complication rate of normal tissues and decreases the dose required at the target. Because the target receives a lower than prescribed dose, tumor control can be adversely affected.

Incorrect placement of the radiation field relative to patient anatomy results in the same effects described above. This misalignment primarily comes from patient movement and repositioning errors. Rabinowitz et al. [Rab85] have analyzed the simulation and port films to determine the field misalignment errors. By using the existing immobilization techniques, the average radiation field discrepancies for consecutive films vary from 3.5 mm to 9.2 mm for the head and neck region and the thorax region. Since the bony structures of the head and neck region could be strongly immobilized, an average error range of 3.5 mm is the minimum field alignment error of the whole body area according to study results. The localization method of stereotactic radiosurgery are potentially useful to assure patient positioning reproducibility during each treatment fraction since the reported accuracy is higher than that of any treatment techniques applied in the generalized routine radiation therapy.

Fractionated High Precision Radiotherapy

Fractionated high precision radiotherapy refers to high precision, multifraction, external-beam irradiation using stereotactic localization methods. In stereotactic procedures, with a single fraction radiation treatment, the prescribed dose can be accurately delivered to the target area. In order to achieve high precision fractionated radiotherapy, it is necessary to develop a technique that allows repeated fixation of patient position for multifractionated radiation delivery schemes. Radiosurgery has been successfully implemented by hardware and software to determine the target size and location, to perform treatment planning, and to deliver proper dose by the plan. However, for certain tumors, the single fraction treatment may provide inferior clinical results when compared to conventional radiation therapy. On the other hand, conventional fractionated radiation therapy may introduce setup errors not present in the stereotactic treatments with only single irradiation delivery. The high precision multifractioned radiation treatment could greatly increase the possibility of local control for various benign and malignant tumors of the brain and skull such as acoustic neuroma, meningioma, pituitary adenoma, malignant glioma, locally recurrent nasopharyngeal carcinoma, and chordoma and chondrosarcoma of the skull base and cervical spine.

Patient repositioning is an important factor to determine success of the tumor control for fractionated treatment. Therefore, the improvement of patient repositioning for fractionated stereotactic radiotherapy is a critical step in achieving better local control.

CHAPTER 2
REVIEW OF LITERATURE

Stereotactic Radiosurgery Systems

Stereotactic radiosurgery has four qualities that distinguish it from routine radiation therapy. They are 1) a target volume that does not lend itself to routine therapeutic simulation techniques, 2) the necessity for true three dimensional treatment planning, 3) stricter isocenter criteria for gantry and patient motions, and 4) small beam size [Bov90a]. Because of the complex structures and critical organs in the brain, high precision and accuracy are critical for treating intracranial diseases. In the 1950s, the Swedish neurosurgeon Lars Leksell built a stereotaxic instrument for open brain operations. He then modified this unit and irradiated the target through a large number of small beams by fixing a semicircular stereotaxic frame to the patient's skull. He named this method radiosurgery. The original radiation source used was a 280 kV X-ray tube [Lek51]. This source of radiation was later replaced by multiple converging cobalt sources, initially 179 and later 201 beams. This device, called the gamma knife, has been in clinical use since 1968 [Lek71].

In 1958 and 1968, other teams in Berkeley [Tob64] and Boston [Kje77] gained experience in the use of charged particles for treatment. These beams of protons or helium ions were produced by synchrocyclotrons, taking advantage of the rapid beam falloff characteristics (i.e., Bragg peaks) to spare adjacent healthy tissues. While these methods are effective, they need very expensive, delicate equipment and are only available in a few large medical or research centers. In 1974 the use of modern linear accelerators (LINACs) as a radiosurgical alternative was proposed [Lar74]. For the past decade clinical studies involving radiosurgery with LINACs have been extensively pursued.

Betti [Bet84] reported the first usage of a LINAC as a standard treatment machine for radiosurgery. Colombo [Col85], Hartman [Har85], Sturm [Stu87], and Lutz [Lut88] subsequently have reported modified LINAC radiosurgery techniques to deliver highly focused radiation to a target. However, most of the groups used the stereotactic reference ring by attaching it to the patient's skull and only delivered a single dose fraction for high precision treatment of intracranial targets. In LINAC-based dose delivery systems, the accuracy and precision of the LINAC gantry and the patient support rotations are generally contained within 2 mm diameter sphere. Since they are seldom concentric, the combined inaccuracy can exceed a ± 2 mm radius sphere [Bov90a]. These errors are significantly greater than those of target localization, and potentially could limit the precision of the procedure. Friedman and Bova [Fri89a] at the University of Florida developed a three dimensional sliding bearing system to couple an auxiliary collimator to the LINAC head. An overall system accuracy of 0.2 mm with a standard deviation of 0.1 mm was achieved. This level of accuracy and precision will permit significant increases in the spacial accuracy of image localization before the dose delivery system becomes the limiting factor in radiosurgery. To date, this is the most precise and accurate system of those described in the literature. It is commercially available as the Philips SRS 200 Stereotactic Radiosurgery System. With this radiosurgery system, a high precision treatment of radiation beams has been successfully applied clinically to irradiate inoperable intracranial targets.

Using a routine simulator, the treatment beam cannot visualize the patient. Routine stereotactic radiosurgery is a true application of virtual simulation in that the mathematical description of the patient is utilized to plan the treatment process. Then the treatment is based on the virtual planning process, in which the radiation beams are manipulated mathematically to irradiate proper patient anatomy. This virtual process with 3-D patient information and Beam's Eye-View [Goi83] function could improve the precision of radiation delivery relative to the correct patient anatomy.

Radiobiology for Fractionated Treatment

In 1914, Schwarz pointed out that a single dose radiation treatment might not be efficient because some of the tumor cells were in different stages of radiosensitivity, and there was a better chance that multiple exposures could hit the cell in a radiosensitive phase [Per87]. However, the basic parameters of dose versus time are complex functions. Moss noted the following advantages of dose fractionation [Mos79]:

(1) There is a reduction in the number of hypoxic cells through cell killing and reoxygenation,

(2) There is a reduction in the absolute number of tumor cells by the initial fractions with the initial killing of the better-oxygenated cells,

(3) Blood vessels compressed by a growing cancer are decompressed as the cancer

shrinks, permitting better oxygenation,

(4) Fractionation exploits the difference in recovery rate between normal tissues and neoplastic tumors, and

(5) The acute effects of single doses of radiation can be decreased with fractionation.

The basic parameters in dose-time considerations constitute a complex function, which interprets the interdependence of total dose, time, and number of fractions to produce a biological effect in the tumor. The fractionation schemes to generate expected biological responses of both tumor and normal tissues are closely related to the four "Rs" of radiation [Hol91, Per87]:

(1) Repair of sublethal damage of potentially lethal damage,

(2) Reproductions of cells between each fraction,

(3) Redistribution of cells throughout the cell cycle, and

(4) Reoxygenation after radiation exposures.

The repair of sublethal damage produces the clinical observation that larger total doses can be tolerated when the treatment is divided into multiple small fractions [Hol91]. This biological factor indicates that the fractionation of radiation therapy could introduce clinical tumor control benefits because the repair mechanisms of normal tissues and tumor are differentiated [Gei87]. A small fraction dose allows normal tissues to recover to a larger fraction of the cell population for the next irradiation, while the less recovered tumor cells continue to be damaged by the following irradiation. Fractionation treatment attempts to deliver radiation therapy beams to achieve the total destruction of the proliferative capability of the tumor cells, while leaving a certain level of function of the normal tissues or organs involved. A linear-quadratic survival relationship (α/β ratio) of the cell responses to radiation is used to assess the fractionated doses (Fig. 2-1). This characteristic curve is represented as follows:

$$\mathrm{Log}_e S = \alpha d + \beta d^2 \tag{2.1}$$

S is the survival fraction of cells; α represents the linear, nonseparable component of log cell kill; and β represents more effective killing after some repair process has been eliminated with increasing dose [Per87].

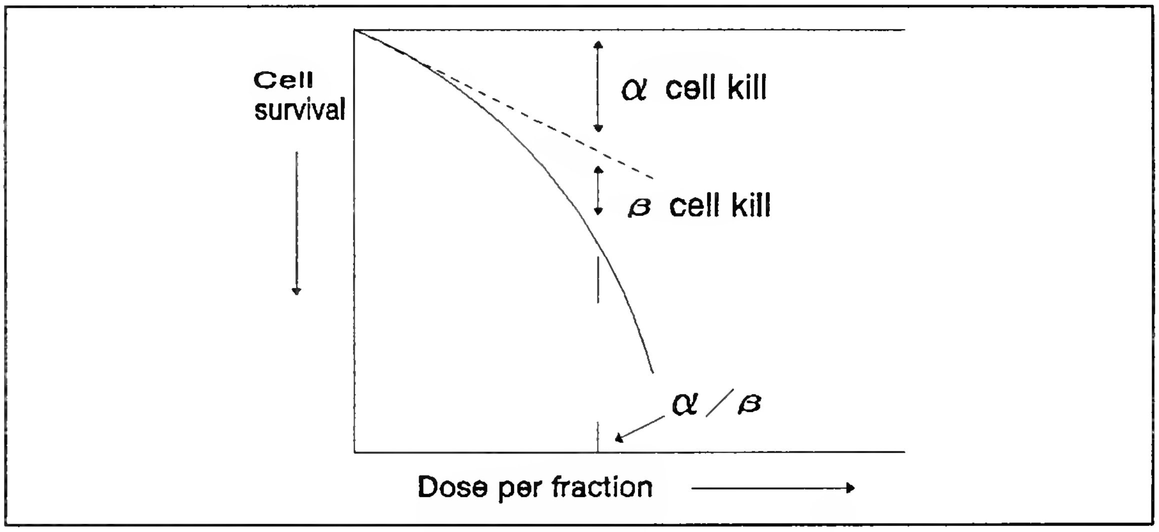


Figure 2-1: Linear-Quadratic model for multifraction dose assessment [Per87]

AVM, for instance, is a late-response tissue with small α/β ratio. Fractionation will not generate benefits for destruction of this late-responding tissues. However, for tumor control, the α/β ratio is larger, which means that the dose fractionation spares late-responding tissues more than early responding tissue. The biological benefits are then observed.

In order to perform a fractionated procedure in the stereotactic environment, it is important to identify the target location repeatedly and precisely. Fractionated treatment of radiotherapy requires reposition of patients for each treatment course relative to a specified coordinate system. Clinically, the error ranges have been extensively examined and evaluated. Many methods have been created to minimize the errors.

Localization Errors

Accurate daily positioning of the patient in the treatment position and reduction of patient movement during treatment are essential to deliver the prescribed dose and achieve the planned dose distribution. In the study of anatomic and geometric precision achieved in the treatment of mantle and pelvis port films, one third of the localization errors were caused by patient movement [Hau72]. It then shows that the difficulty of positioning the customized shielding blocks is caused by the movement of the external landmarks on the surface of the patient, which are used to indicate the correct positioning of the shields. When the landmarks move, the shields move in relation to the internal anatomy, and localization errors occur [Mar74].

Hodgkin's disease provides a difficult metastasis site with resulting frequent field placement errors. Marks et al. reviewed the verification films to detect geometric miss or localization error and found that the maximum number of 44 errors resulted from a total of 77 verification films during one study period. A maximum of 55% of the localization error has been observed [Mar74]. Those localization errors result from the external landmark changes because of movement of skin and subcutaneous tissues over underlying skeleton and musculature [Mar76]. Rabinowitz et al. studied the accuracy in all radiation fields alignment in clinical practice. They found that the mean worst-case discrepancy averaged over all treatment sites was 7.7 mm, the lowest mean value was 3.5 mm in the head and neck region with proper fixation, and the highest mean value was 9.2 mm in the thorax region [Rab85]. Errors of over 1 cm were found in 10% to 23% of the radiation fields and an increase of in-field relapse rate from 7% to 33% was found, when the radiation beams did not encompass the tumor region adequately [Hul89].

Byhardt et al. [Byh78] separated field misplacement into four categories: field malposition, field malrotation, patient malposition, and block malposition. An overall field placement error was 15% (10% if the error limit was set at 1.0 cm or greater). Field malposition, which means cephalo-caudad and/or transverse field shift, was the most common field misplacement and composed 74% of all errors.

Goitein [Goi86] summarized the reasons for nonuniform dose delivery and concluded that some degree of field misplacement is virtually inevitable. He then suggested a margin of approximately $1.5\times\sqrt{\sigma_m^2+\sigma_p^2}$ where σ_m is the standard deviation of patient motion and σ_p is the standard deviation of the penumbra of the dose profile.

According to clinical surveys in routine radiation therapy, several millimeters up to centimeter range inaccuracy in relocalization are observed for fractionated treatment. Those inaccuracies increase normal tissue complications as well as decrease tumor control.

Immobilization Methods

The precision required in radiation treatment is most critical in management of tumors in the head and neck area where the proximity of the optical nerve structure, brain, and cervical spinal cord necessitates a high degree of precision in both patient positioning and immobilization. However, to improve treatment accuracy, many repositioning and immobilization techniques have been created. Khan [Kha84] mentioned some general guidelines for maintaining the patient treatment position, such as a bite-block system, a head clamp and mask for head and neck treatment, and a partial body mold or cast for irradiation of other body parts. Choice of the technique depends on the location of treatment fields.

Williamson [Wil79] developed a styrofoam block for defining posterior landmarks in position on a simulator to improve the lateral treatment accuracy. Verhey et al. [Ver82] developed a number of immobilization schemes that permit precise daily positioning of patients for radiation therapy. They studied different patient positions that indicate the movement from a mean of 1.4 mm for different treatment areas. A daily reproducibility of patient position using laser alignment and pretreatment radiographs for final verification of positions indicates that the initial laser alignment can be used to position the patient within 2.2 mm $\pm$ 1.4 mm. By using a combination of pretreatment radiographs and rather simple contoured plastic masks or casts, holding intracranial treatment movement to less than 2 mm is possible in over 80% of the treatment studied. This result indicates that rigid immobilization devices can improve the radiation therapy, especially in the head and neck regions [Ver82].

Thermal plastic (Aquaplast™) has been employed for head and neck treatment because of its low cost and easy utilization [Ger82]. In clinical practice, physicians tend to use line marks and thermoplastic together to avoid possible treatment field localization errors. Custom molds constructed with the use of polyurethane formed to patient contours are also becoming major aids to immobilization and repositioning. Some typical sites, including breast and upper body for Hodgkin's disease, need molds for strong support of body structures. Another immobilization device is the vacuum-form body immobilizer, which is often used for rigid body positioning and easy adjustment during the treatment schemes.

Some institutions developed an elaborate casting technique to immobilize and reposition patients during radiation treatment. This technique is not popular because there is a potential for movement inside the cast due to patient weight loss or the regression of tumor. All those techniques try to reposition the patient properly; however, they rely on individual skill and differ from institution to institution. As discussed in the previous section, utilization of these immobilization techniques results in an average minimum inaccuracy of 3.5 mm, which means there is room for improving the treatment precision.

Fractionated Stereotactic Radiotherapy Systems

As mentioned previously, stereotactic radiosurgery is a high precision radiation therapy technique to treat intracranial disorders. Invasive and single fraction radiosurgery has been proved an effective methodology to treat inoperable brain diseases. However, in order to differentiate tumor dose response with normal tissues, fractionated stereotactic radiotherapy may be the preferable choice. Therefore, patient repositioning with a high degree of accuracy to spare critical structures inside the brain becomes the ultimate goal. Stereotactic radiotherapy requires a high degree of precision because of the very steep dose gradients used, typically decreasing the dose from 90% to 50% in

2 mm. The localization errors from several millimeters to a centimeter in routine clinical applications are inappropriate for this type of treatment, especially due to the close proximity of critical structures. For example, routine radiation treatment with a field size of 15 cm x 15 cm and a ±2 mm error margin is mostly possible and will not greatly increase the complication rate in most body sites. However, with a small treatment field of 2 cm x 2 cm and a ±2 mm error margin near the critical structure such as the brain stem, this error might cause complications simply because the penumbra region of the beam profile might cover part of the critical organ. Fractionated stereotactic radiotherapy is initiated to achieve two goals: delivering the highest accumulative tumor dose to the target area throughout the treatment course and minimizing the beam portal errors by accurate localization of the tumor with respect to the treatment beams.

Refixation Techniques

To perform fractionated stereotactic radiotherapy, several fixation techniques had to be developed. The easiest method is to use a special head frame or mask for immobilizing the patient's daily treatment position. Lyman et al. [Lym89] developed a mask and stereotactic frame combination technique for charged particle radiation treatment. Usually in clinical environments facial or bony landmarks are matched to correct possible relocalization error. The mask has to be made giving particular attention to the bony structures such as the mandible, zygomatic arches, frontal ridges, and nose. The mask is divided in two halves coronally and requires careful molding procedures. As Lyman discovered, the mask is fit over the skin and hair of the head and relies on friction for immobilization; it is found that some slippage can occur for large rotations around the localization position because the proton beam can only be delivered at a fixed angle. Thus, the port angles of beams will be limited in order to prevent the large rotational errors of the patient. The immobilization of the patient's head in a reproducible manner within the stereotactic frame is based on the visualization of landmarks, for instance, nose, chin, and frontal processes. The proper fit of the mask will be under continuous evaluation from procedure, to procedure which might require extra time for patient immobilization [Lym89]. With the system, they reported an error of approximately 1 mm in repeated sessions in each of three orthogonal directions could be achieved. This is only the repositioning error, not the total treatment error.

Another factor influencing patient position is the potential for patient movement during approximately 1 hour of diagnostic procedures. This effect will distort the coordinates that were registered in the images. For stereotactic radiosurgery, which needs high treatment accuracy, using only a mask without other supporting devices seems to have some potential problems.

Hariz and his coworkers [Har90] designed the Laitinen® stereoadapter for fractionated radiotherapy. The noninvasive stereoadapter shows a high degree of reproducibility between repeated mountings. The total maximum error between isocentrical positionings of the target does not exceed 3 mm.

Delannes et al. [Del91] uses Laitinen's stereoadapter to immobilize the patient's head. This non-invasive head frame depends on the patient's facial and head structures. A 1 mm spatial coordinate precision of reproducibility is reported. Gill et al. in UK use a noninvasive relocatable frame to localize patient position by mechanically and physically constraining the patient skull for stereotactic external beam therapy [Gil91]. This frame is also employed for fractionated radiotherapy, with overall mean displacements of 1.2 mm for AP and 1.8 mm for lateral directions [Gra91]. These measurements are repositioning inaccuracies; they do not include any localization errors. However, the reproducibility of the positioning of the head for each of these procedures depends on the closeness of the fit of the stereoadapter, the uniqueness of each patient's cranial and facial features, and the cooperation of the patient. The amount of soft tissue and the uniqueness of cranial features are related and greatly affect the reproducibility of positioning [Lym89]. Jones [Jon90] applied three implanted gold markers to define the reference coordinate system, which promoted fractionated therapy. The total potential misalignment of this method in all directions could be up to 2 mm.

Relocalization Methods

Pelizzari and Chen [Pel87,89,91] developed a method to allow accurate registration of different image modalities (CT, MR, PET). Utilizing a nonlinear least-squares search, the "hat" image (image to be transformed) is translated and rotated to minimize the differences from the "head" image (reference image). This is the so-called "Hat" and "Head" method. From 3-D reconstructions of the patient's face and skull, data points that are far apart can be used to match these two rigid bodies. An error range of 1 mm is generated by careful manipulation of the digitizer system. This method could be applied to reposition the patient's head for stereotactic radiotherapy. Using a three dimensional digitizer, the relative coordinates of the new patient position can be obtained with respect to the reference position. Thus the localizer marks on a patient head can be realigned to match treatment room laser planes for gantry and treatment couch setups.

Kato et al. [Kat91] have developed a frameless, armless neurological system to localize the target inside the brain. They used a 3-D digitizer with low-frequency magnetic field to detect the 3-D coordinates and the three orientation angles. Mean positioning errors from 1.7 mm up to 4.0 mm were reported when the digitized probe was tested. However, when the magnetic digitizer system interferes with metals, coordinate shifts as much as several centimeters were observed. This system can locate the intracranial target with careful manipulation. In comparison with the conventional head frame systems, the accuracy is still under investigation.

Barnett et al. [Bar93] utilized a frameless, armless stereotactic localization system in brain tumor surgery. The localization system consists of a localizing wand that emits ultrasonic pulses that can be detected by a table-mounted array of microphones. With the triangulation of the emitter signals, the 3-D localization of the wand tip can be determined with respect to the microphone detector arrays. In the operating room environment, using four spark pairs, the reproducibility of a point in space is $\pm$0.6 mm standard deviation. The mean error for measuring distance within a 1000 cm cube is 1.1 mm $\pm$ 1.0 mm, which represents 1.0% $\pm$ 0.7% in space. However, for the overall determination of accuracy, the mean linear error in localizing 19 targets in 48 patients is 4.8 mm $\pm$ 2.1 mm standard deviation. This error term refers to the image data of 3 mm thickness computed tomograph or 2 mm voxel magnetic resonance imaging volume acquisition.

As previously described, researchers in other clinics used landmarks to properly reposition the patient's treatment position. Those landmark positions were retrieved from visual localization or by using plain radiographs. Errors can be analyzed by using radiographs or a portal image system [Mee90]. All the fractionated treatment courses are based upon the error estimation of the treatment position. Therefore, the relocalization procedure becomes a critical issue to realign the tumor for fractionated treatment.

All the fractionated radiotherapy methods have potential patient repositioning errors. Errors from millimeters to centimeters have been discussed previously and are based on the observed results. Those methods need to pay particular attention to bony landmarkers; experienced manipulation is required to correct possible errors. The drawbacks of those repositioning systems are prolonged procedures which introduce potential patient movements. For high precision fractionated stereotactic radiotherapy, a fast, accurate, and easy technique would be the ideal alternative.

CHAPTER 3
THE UF STEREOTACTIC SYSTEMS

The UF Stereotactic Radiosurgery System

Hardware Characteristics

A modified radiosurgery system of the BRW™ stereotactic ring (a conventional neurosurgery device) has been designed at the University of Florida and commercialized as the Philips SRS 200 radiosurgery system [Bov90a]. It is a three gimbal bearing system; the head stand and bearing-holder system are incorporated into a single portable piece of equipment (Fig 3.1). Instead of attaching the BRW ring to the treatment couch, it is attached to the head stand, which delivers better accuracy in the patient's treatment position. This head stand provides high accuracy micrometer adjustment in positioning the localized tumor target center to coincide with the system isocenter.

The swing arm system provides rigid rotation and support of the auxiliary collimators. The auxiliary collimators are to define the treatment beam area and minimize the penumbra effects. The set of circular collimators ranges from 5 mm to 40 mm in diameter and with proper divergence delivers superior treatment volume selection and optimization.

When the heavy LINAC gantry head rotates towards the horizontal treatment position, the gantry sags and tilts, displacing the central beam below the isocenter. A gimbal bearing system is designed to decouple the torque of the gantry from the auxiliary

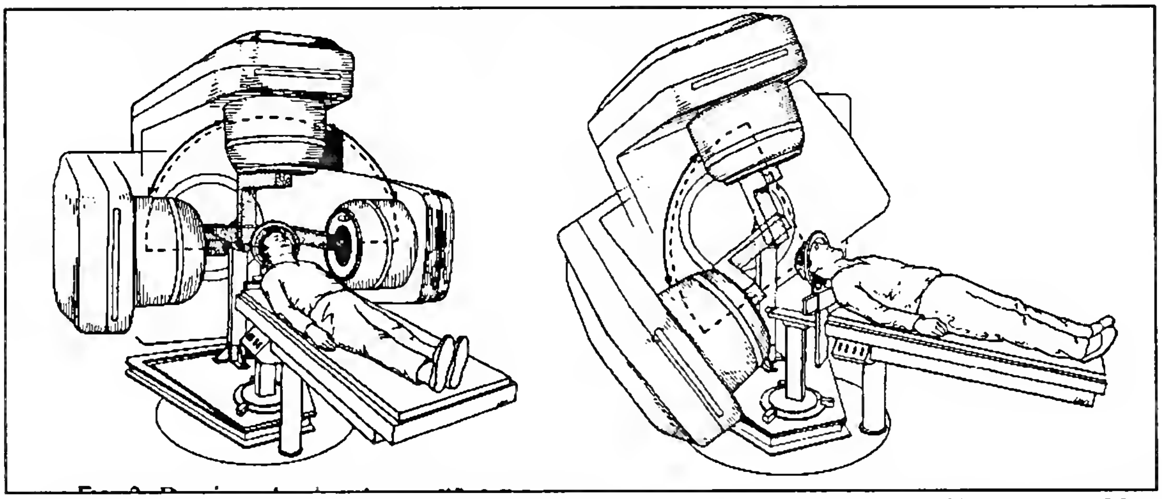


Figure 3-1: University of Florida Stereotactic Radiosurgery System ([Fri92a], page 834, used with author's permission)

collimator and ensures that the sag in the LINAC does not result in angular deviation of the collimated beam from the target point [Fig 3-2, Fri92a].

Treatment Planning Procedures

The stereotactic radiosurgery procedures are currently performed on an outpatient basis. Before starting the diagnostic procedures (Computed Tomography, CT; angiography; or Magnetic Resonance, MR), the BRW ring has to be fixed to the patient's head under local anaesthesia. CT is the standard imaging modality for target localization; if uncertainty exists, angiograph or MR are utilized to visualize the target shape. All the localization procedures refer to the BRW coordinates, which are fixed with respect to the patient's internal and external anatomy. After finishing the imaging sequence (CT or MR), these images are transferred to the treatment planning system (Sun™ Workstation) and the localization process starts. If an angiograph has been performed to locate the target, the biplane films with sixteen fiducial markers (eight per

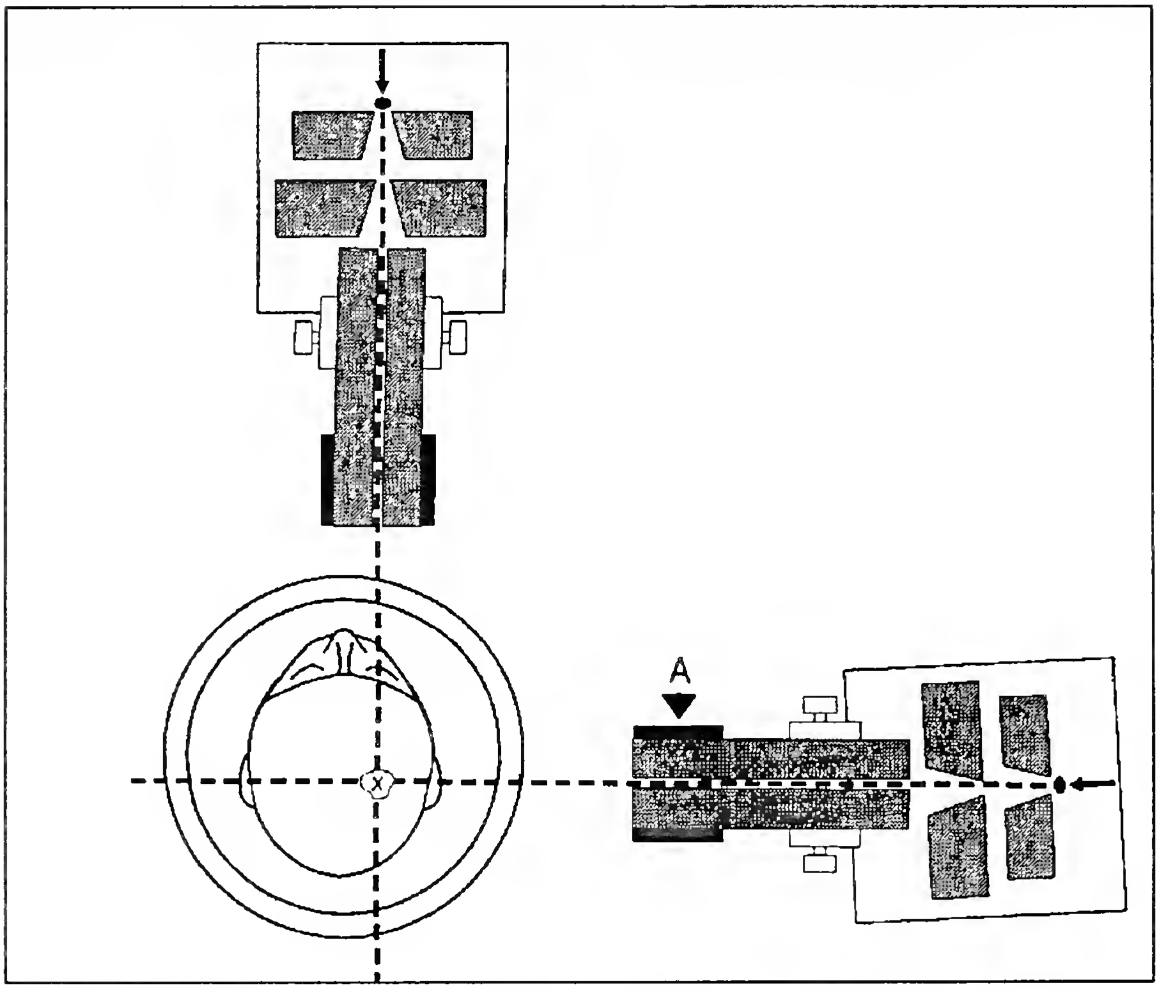


Figure 3-2: LINAC gantry sag was taken care of by the gimbal bearing (A) system, therefore the mechanical integrity of 0.2 mm ± 0.1 mm is maintained. ([Fri92], page 146, used under the author's permission)

AP/lateral projection) are used to define the reference coordinates [Sid87]. Then the radiopaque markers are digitized into the treatment planning computer for the calculation of a geometrical center and a center of mass that closely matches the nidus [Bov91]. The best matching pair should be used as the center point of target.

A state-of-the-art treatment planning software was developed to perform rapid dose calculation and to display isodose distributions. A standard nine arc plan, shown in Table 3-1, is the starting plan for the final optimized plan. Varying such factors as table angles, collimator sizes, or beam weighting parameters changes the isodose distribution on three orthogonal views. Multiple isocenters are specified if the target is extended or irregularly shaped. The visual optimization of the treatment plan is considered a key factor in obtaining the best plan, since it has to take sophisticated planning interaction into consideration.

Patient Treatment

After the best treatment plan has been obtained and the adequate treatment information is collected, the stereotactic radiosurgery accessories are then attached to the LINAC and the target isocenter is set on the head stand. Setting up the UF radiosurgery system takes less than 15 minutes. An independent phantom is then set up to match the isocenter position, and x-ray images of the phantom target locations are taken at various table and gantry combinations [Win88]. If the images show target position in the center of collimator field within ± 0.2 mm, the head stand is considered at the right treatment position.

The patient is brought to the room, and the BRW halo is attached to the head stand. Treatment then proceeds as defined in the treatment planning procedures. At the end of treatment, the BRW ring is removed and patient is free to leave.

The UF Fractionated Stereotactic Radiotherapy System

Hardware Characteristics

The University of Florida stereotactic radiotherapy system has the following goals:

Table 3-1: Standard nine arc treatment plan

Arc Number	Collimator Size	Table Angle	Gantry Start Angle	Gantry Stop Angle	Arc Weight
1	10 mm	10°	30°	130°	1
2	10 mm	30°	30°	130°	1
3	10 mm	50°	30°	130°	1
4	10 mm	70°	30°	130°	1
5	10 mm	350°	230°	330°	1
6	10 mm	330°	230°	330°	1
7	10 mm	310°	230°	330°	1
8	10 mm	290°	230°	330°	1
9	10 mm	270°	230°	330°	1

(1) It should be frameless. The traditional BRW ring system in the LINAC radiosurgery provides rigid coordinate transformations that bring the tumor center to the machine isocenter. The BRW coordinate system is not valid for fractionated treatment if the ring is removed. Instead of the use of invasive pins on the patient's skull, the patient will be held in position with the aid of four soft rubber head posts and soft pressure pad systems. To minimize the rotational error corrections, the diagnostic procedures (CT, MR, and angiography) will be carried out with the patient positioned in the same immobilization device that will be used for treatment. The head holder for CT and MR has a convergent set of rods embedded into the base and provides a method whereby the axial increment of the CT or MR coordinate can be verified by means other than the scanner readouts. The head holder also allows the attachment of two orthogonal planes with fiducial markers, permitting the reconstruction of the target position as described by Sidden et al. [Sid87].

(2) It should achieve the highest treatment accuracy. In order to obtain the highest accuracy of the UF refixation technique, this device will be incorporated into the existing UF radiosurgery system. Since the UF radiosurgery system provides the best mechanical treatment accuracy among the LINAC-based radiosurgery systems, this add-on refixation device will deliver the highest treatment accuracy.

(3) It should correlate to the tumor shape as closely as possible. Experience shows that many tumors are irregularly shaped. By simply translating the patient with respect to the laser or couch, readings for fractionated therapy may cause target misalignment either in a linear direction or in orientation. To overcome the problem and

to correlate the tumor shape as close as possible for different treatment courses, a six degree-of-freedom parameter estimation process is necessary to realign the patient for correct treatment position and orientation. Algorithms for patient realignment with translation and rotation operations have been developed as part of this work. This software provides keys to the treatment of patients with intracranial abnormalities. A mechanical device will be created to provide the necessary mechanism for repositioning purpose. Therefore, not only the isocenter treatment position can be properly aligned but also the target shape can be matched with the minimum normal tissue exposed to the high precision, high isodose gradient treatment.

(4) It should be easy to use. Since a lengthy treatment preparation procedure may cause potential patient discomfort and movement, this system should be easy to adapt to the UF radiosurgery system and should not be complicated to operate. Setup time of this device should be short so that the patient can tolerate the treatment without difficulty. Daily setup should also be easy for the technologist to achieve optimum treatment and to do so economically.

Localization and Refixation Procedures

The procedure of frameless repeat fixation and treatment starts with a dental bite plate system. This custom bite plate will be fabricated by a prosthodontist with three fiducial markers in place for later verification purpose. This procedure is performed prior to the diagnostic scans. After the bite plate system is placed in the correct position, the patient will go through the CT or MR scanning process. Each image not only contains the 3-D information of the patient's anatomy but also the coordinates of the fiducial markers relative to the 3-D image data sets.

After the patient scanning procedures are done, a similar treatment planning session as described in the UF radiosurgery papers [Fri89a, Bov90] is performed to calculate dose distribution of multiple non-coplanar arcs. The dose to the target is then maximized while the dose to the normal tissues and critical structures is minimized. The position of isocenter(s) could be obtained as well as those of fiducial markers.

The patient is then positioned at the LINAC with the UF radiosurgery system for strict isocenter control. The patient is immobilized with the soft pad head-clamping system which allows little or no head movement during the treatment procedure. This immobilization device should not interfere with any fiducial markers and anatomical reference points. Then the fiducial markers are digitized with a 3-D camera digitizing system which is well calibrated with respect to the machine isocenter. Software programs are executed to obtain translation vectors and rotational angles required to realign the patient's treatment position. A 3-D translation and three-axis rotation system is designed and adapted into the existing University of Florida radiosurgery system as a subsystem to perform the patient realignment function.

Once the patient is properly repositioned according to the calculation results, the modified 3-D image correlation program which was originally initiated by Pelizzari/Chen [Pel91] at the University of Chicago or the new programs developed at the University of Florida can be used to compute and ensure the final treatment coordinates by using a portable personal computer. After this verification procedure is finished, treatment with a regular stereotactic treatment plan may proceed.

Patient Treatment

After the patient treatment position is "fine tuned" to the correct coordinates, a verification system consisting of three spheres will be employed. The current location of these spheres, which are incorporated into the basic CT/MR data sets are then verified prior to treatment. The bite plate system differs from others used in that it is not used for repositioning and does not have any realignment pressures applied, thereby maintaining its true geometry relative to the patient's anatomy. The fiducial markers are then digitized again to ensure that the final coordinates are valid. The patient treatment position actually can be verified at any time when there is suspected patient movement. If minor errors exist, a decision can be made as to the adjustment or relocalization of the treatment target position. The daily treatment will start and the radiation is then delivered. Since the final position of the fiducial markers on the bite plate system has the greatest sensitivity on the treatment position errors, this verification system will certainly provide the greatest treatment accuracy.

A plain film procedure can also be utilized to verify the final treatment accuracy if necessary. This procedure would provide yet another independent technique for the fractionated treatment schemes.

CHAPTER 4
TARGET RELOCALIZATION BY 3-D CONTOUR POINTS

Statements of Problems

Radiosurgery can achieve extreme accuracy by immobilizing patient's position with strong fixation (i.e. stainless steel pins into the skull). Single fraction treatments are designed to generate maximum damage to the target with high precision. However, they cannot take advantage of the maximum benefits of fractionated treatments. Some tumors, such as a large pituitary tumors or craniopharyngiomas, are more sensitive to radiation than are normal brain tissues when multifractions of radiation are delivered. Therefore, fractionated, stereotactic radiosurgery would be the ideal treatment choice.

While treating intracranial tumors, traditional radiosurgical treatment might place the optical nerve at risk, as tumors may directly abut that structure. Utilization of routine radiation therapy techniques would involve large fields, encompassing much more normal brain tissue than would utilization of radiosurgery techniques. An ideal compromise would be the fractionated, stereotactic technique [Spi91]. The traditional BRW stereotactic frame and other fixation frames fix directly to the skull by means of pins and anchor posts. Several new stereotactic frames have been developed which locate to the bony anatomy and are acceptable for repeat fixation for fractionated therapy [Gil91, Har90, Hou91, Jon90, Lym89]. To achieve better tumor-dose response and to reduce dose to the normal brain tissues, the fractionated technique is very important for tumor treatments other than AVM.

The basic principle for fractionated treatment is to reposition the target isocenter back to the machine isocenter for each treatment fraction. Registration of the patient's contours or the fiducial markers is an important step in realigning for repositioning. Most systems perform repositioning only by repeating the treatment settings from laser or treatment couch readings [Gil91, Hei89]. Due to limited accuracy of the devices, this might put some of the critical structures at risk. In order to fully realign the treatment position, not only is the translation back to the isocenter necessary but also the 3-D orientation of the tumor should be matched. Therefore, with six degree-of-freedom fitting procedures (translation and rotation) instead of using three degree-of-freedom parameters (translation only), the tumor shape can be matched as closely as possible.

Correlation of two sets of data points is a many-to-one mapping problem mathematically. Calculation of translational and rotational parameters is done by solving nonlinear equations. Theoretically, no analytical solution is available for these equations. A least-squared approach is the favored choice for solving this kind of problem. Since all the 3-D data points contain certain types of digitizing and systematic errors, finding the best fit of these two different 3-D data sets is also equivalent to minimizing the error terms.

Verification of the treatment position is also an important factor in identifying the correct treatment geometry. A separate system should be supplied to perform independent treatment verification. Error analysis of the 3-D data sets and the target position and orientation should also be performed.

Methodology

The methodology for patient repositioning at the University of Florida is based upon the calculation of six degree-of-freedom parameters of two registration data sets which come from different fractionated schedules. Mapping of those anatomical points and contour lines becomes the key issue in calculating the patient's realigned treatment location. Most of the researchers only perform the translation function to realign the patients. According to the references [Del91, Gil91, Har90, Har85, Hei89, Jon90, Pel 91, Tho91], the treatment precision ranges from 1 mm to 5 mm which depends upon the accuracies of laser and treatment couch, the patient's motion, and the operator's skill. In order to correlate the "true" tumor position and orientation, rotational parameters are necessary to fully realign the target, especially in the 3-D orientation of the irregular target shape.

Kessler et al. [Kes91] has described four different methodologies to implement the calculation of transformation matrix for two different 3-D data sets. They are (a) point matching, (b) line matching, (c) surface matching, and (d) interactive matching. In the point matching method, the transformation parameters are determined by establishing a one-to-one response between a set of point landmarkers visible in both fractionated treatments. These points can be either internal anatomical landmarkers or externally placed fiducials. Internal landmarkers include structures such as optical nerve foramen, the center of orbits, and the odontoid process, etc. External markers depend on the image modalities and selected fiducials. For the study, the external markers are arranged in a fixed pattern for repeat localization. In situations where one of the image studies covers only a limited extent of the cranium, it is difficult to place fiducial markers to ensure the visibility. Line pattern markers are suitable for the continuous scan which can provide the location on each image data set. The line markers can also be marked at a specified position which correlates the image coordinates.

Surface matching method uses the edge-thresholding algorithm to extract the outer contour lines from the geometrical feature of patient anatomy. Since the patient head is a rigid structure, the contour information should be invariant over time. The head model is created by forming a pseudo-solid model through the triangulation of the contour lines. The transformation is solved by minimizing the mismatch between two surfaces, which is so called the head/hat method by Pelizzari/Chen [Pel89, 91]. The transformation matrix is achieved by manipulating the hat (by varying the transformation parameters) to properly fit (by minimizing the mismatch) onto the head model. This method provides six degree-of-freedom fitting of two data sets. However, the precision for minimizing the mismatch is affected by the inherent tolerance of the hardware system and the computing work load is heavy. The complexity of determining the transformation parameters is proportional to the multiplication of the number of triangles formed on the head model and the data points on the hat model.

In the interactive matching technique, the transformation parameters relating two 3-D data sets are determined manually using a graphics utility. The method also needs a relatively high computing power to assure that the information can be retrieved and displayed during a short period of time.

We intend to employ both the anatomical points and the patient surface contour line to perform and estimate the mapping function. A vertical facial contour line of patient midplane is selected as well as the anatomical points are recorded to form a complete data set for patient texture information. This data set will be used to correlate the corresponding points or line with proper identification on CT/MR images. In order to correctly set the treatment coordinates for a patient's fractionated treatment, several algorithms have been developed to identify the six degree-of-freedom fitting parameters. These algorithms will be compared with the head/hat concept and identify the validity of the calculations. The following approach explains the procedure for completing the calculation task in finding transformation parameters.

First the patient's midline is digitized as one reference data set; secondly several obvious anatomical points are also digitized as part of the mapping data set. Those data points are integrated as one 3-D reference set of points, which represent the 3-D structure with respect to the patient's own coordinates (this coordinate system will be the same as that of the LINAC treatment coordinates). Fig. 4-1 shows the conceptual drawing of registration of the 3-D contour points.

Since the CT/MR data sets have finite resolution of the pixel size and image scanning slice thickness, the data sets always contain a discontinuous step function for each pixel value. Therefore, a spline fitting function is necessary and is incorporated to resample the midline contour for data mapping purpose. A piecewise cubic spline fitting function is applied to obtain the smooth and continuous curve for the patient's midline. The spline fitting procedure will be considered as follows [DeB78, Wol92]: the 2-D

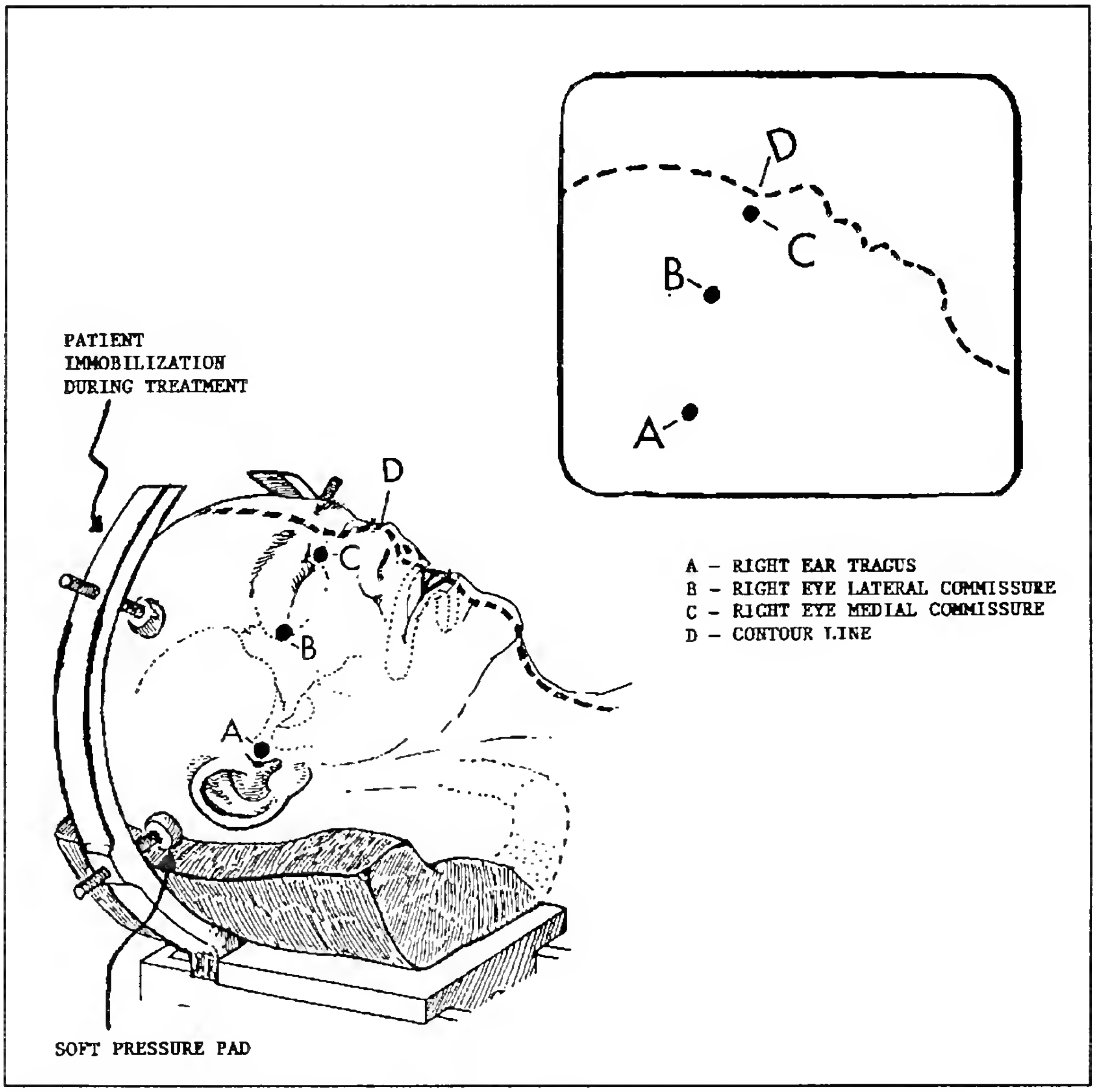

PATIENT IMMOBILIZATION DURING TREATMENT
SOFT PRESSURE PAD
A
B
C
D
A - RIGHT EAR TRAGUS
B - RIGHT EYE LATERAL COMMISSURE
C - RIGHT EYE MEDIAL COMMISSURE
D - CONTOUR LINE

Figure 4-1: The conceptual drawing of patient contour and anatomical points registration process (courtesy of Dr. Bova)

spline fitting procedure is first selected because of the inherent simplicity in data fitting process. Given a 2-D point sets of (x_k, y_k) for $0 \leq k < n$, then the interpolating cubic spline consists of n-1 cubic polynomials. They pass through the supplied points, which are known as the *control points*. We now derive the piecewise interpolating polynomials. The kth polynomial piece, f_k, is defined to pass through two consecutive input points on the ranges $[x_k, x_{k+1}]$. Furthermore, f_k is joined at x_k ($k=1,\ldots,n-2$) such that f'_k (first derivative of the function) and f''_k (second derivative of the function) are continuous. The interpolating polynomial f_k is then given as

$$f_k(x)=a_3(x-x_k)^3+a_2(x-x_k)^2+a_1(x-x_k)+a_0 \qquad (4-1)$$

The four coefficients of f_k can be defined in terms of data points and their first and second derivatives. If the end point derivatives are not known, the "not-a-knot" condition is assumed to imply that the first and the last interior point knots are not active for calculation [DeB78]. These two knots have the same derivatives as those of their neighboring knots.

By applying the spline fitting function a nice and smooth contour curve can be generated to truly represent a patient's midline structure. The 2-D concept of a spline can be extended into 3-D coordinates. The contour data points in the CT/MR images can be resampled again as the comparative standard for fractionated treatments. The logical steps to form the data set of anatomical points and contour lines are as Fig. 4-2.

A set of computer programs were written to perform data sorting, resampling and reweighting of the initial 3-D data set. After the 3-D reference data set is saved as well as the treatment planning information, a separate 3-D data set which follows the same contour lines and anatomical points is acquired from the 3-D digitizer for mapping. Six degree-of-freedom fitting parameters are then calculated and compared with different algorithms in order to find out the best correlation of these two 3-D data sets.

A real time 3-D digitizer is incorporated into the data acquiring system for the fractionated treatment schedules. This system is well calibrated with respect the LINAC

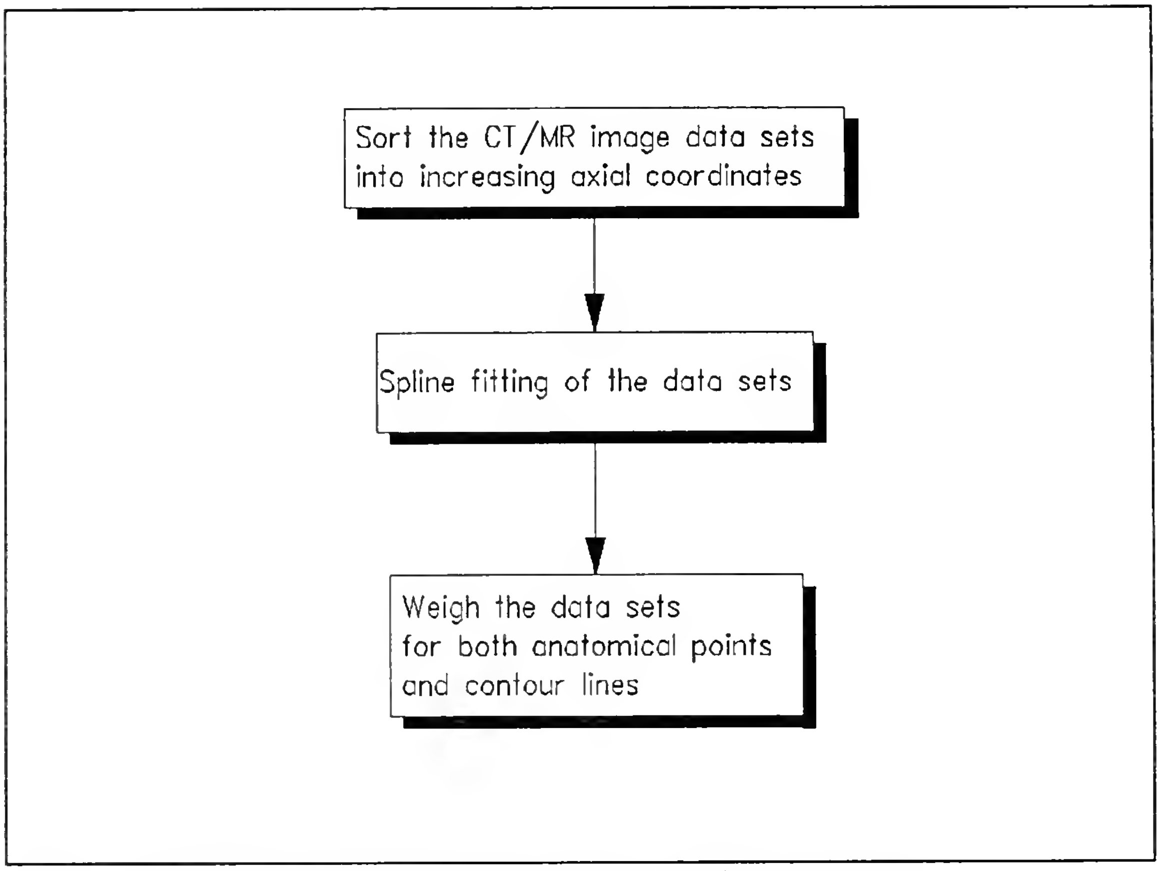

Sort the CT/MR image data sets into increasing axial coordinates
Spline fitting of the data sets
Weigh the data sets for both anatomical points and contour lines

Figure 4-2: Sequence for generating the reference CT/MR data sets

coordinate system. Because of the high resolution and precision, the data sets from this camera system provide reliable information which has the minimum perturbation of the system accuracy. A flow diagram sketched in Fig. 4-3 is presented to integrate various steps for the 3-D data correlation process.

An immobilization device is also created to maintain patient's treatment position while the radiation is being delivered. This add on device has the function to tune in the correct coordinates for the fractionated treatments, remembering that this device cannot interfere with the treatment fields. Therefore, the dosimetry considerations will be the same as that in single fraction radiosurgery. After the patient has been properly

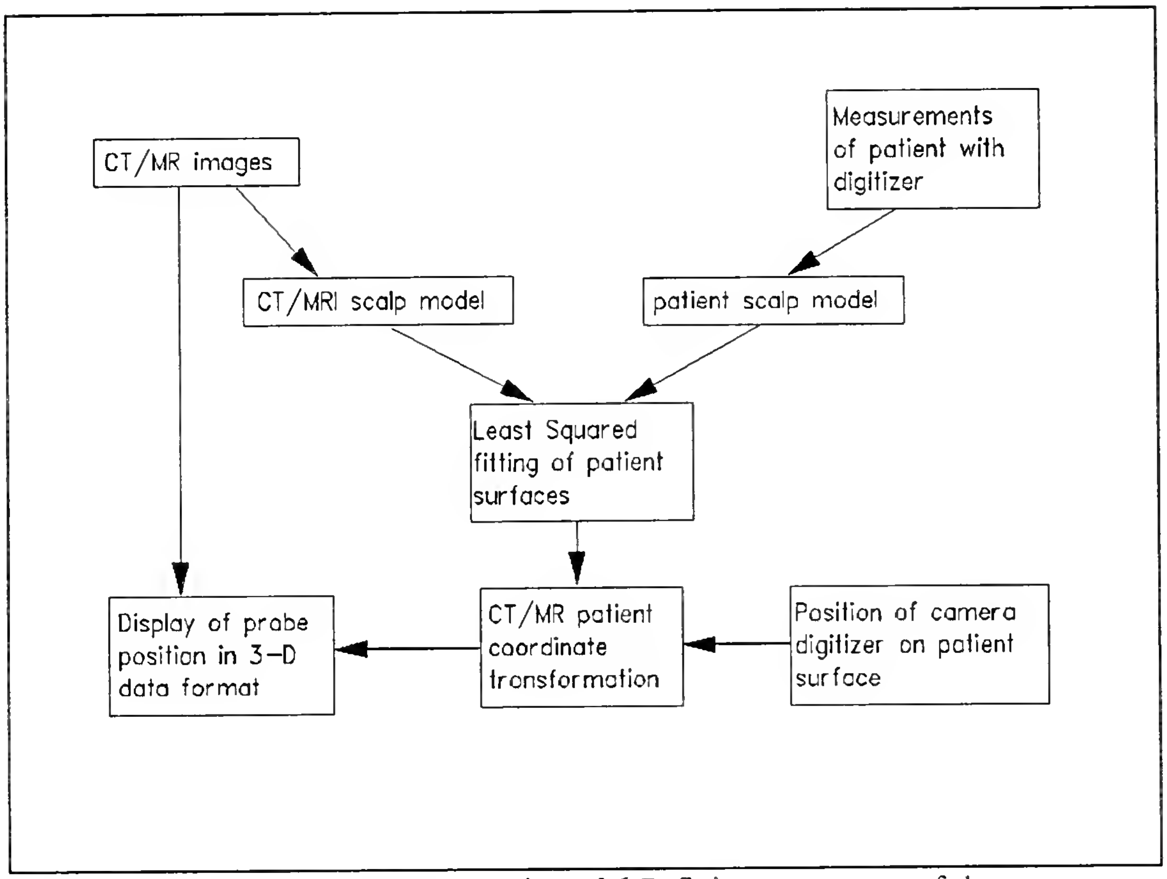

CT/MR images
Measurements of patient with digitizer
CT/MRI scalp model
patient scalp model
Least Squared fitting of patient surfaces
Display of probe position in 3-D data format
CT/MR patient coordinate transformation
Position of camera digitizer on patient surface

Figure 4-2: Procedures for the calculation of 6-D fitting parameters of the Fractionated Stereotactic Radiotherapy

realigned and immobilized, a separate verification bite plate system consisting of three spheres will be employed. The correct location of these spheres were previously incorporated into the basic CT/MR imaging data sets prior to treatment. This system differs from the other systems used is that it is not utilized for repositioning and does not have any realignment pressures applied. Therefore, the true geometry relative to the patient's anatomy (i.e., tumor location) can be maintained. Once the patient's treatment position and orientation are verified as desired, fractionated treatment can start by using the calculated treatment planning information.

CHAPTER 5
3-D COORDINATE MEASURING SYSTEMS

Laser Tracking Interferometer System

The laser interferometer uses light interference principles to precisely measure the linear displacement or velocity of a body. A laser produces a beam of coherent, monochromatic light that is passed through a beam splitter. Part of the beam is reflected towards a fixed mirror and the other part of the beam is transmitted through the beam splitter toward a moving mirror. The light from both the fixed and moving mirrors is reflected back toward the beam splitter, which is designed to recombine the beam as illustrated in Fig. 5-1. The photodetector of the detection system will sense an alternating intensity. One cycle of intensity change represents a mirror displacement of one-half a wavelength of the light from the laser.

The accuracy of the interferometer is affected by any influences that alter the wavelength of light. Therefore, any errors caused by the interference should be properly corrected or compensated. If properly constructed, the laser interferometer could be used to measure 3-D coordinates in space. Systems that combine laser interferometers, optics and electromechanical devices to perform coordinate measurements have been developed by several research centers [Bro86, Lau85, Zhu92]. They use both length and angle measurements to determine the target location. Relative distance measurements provided

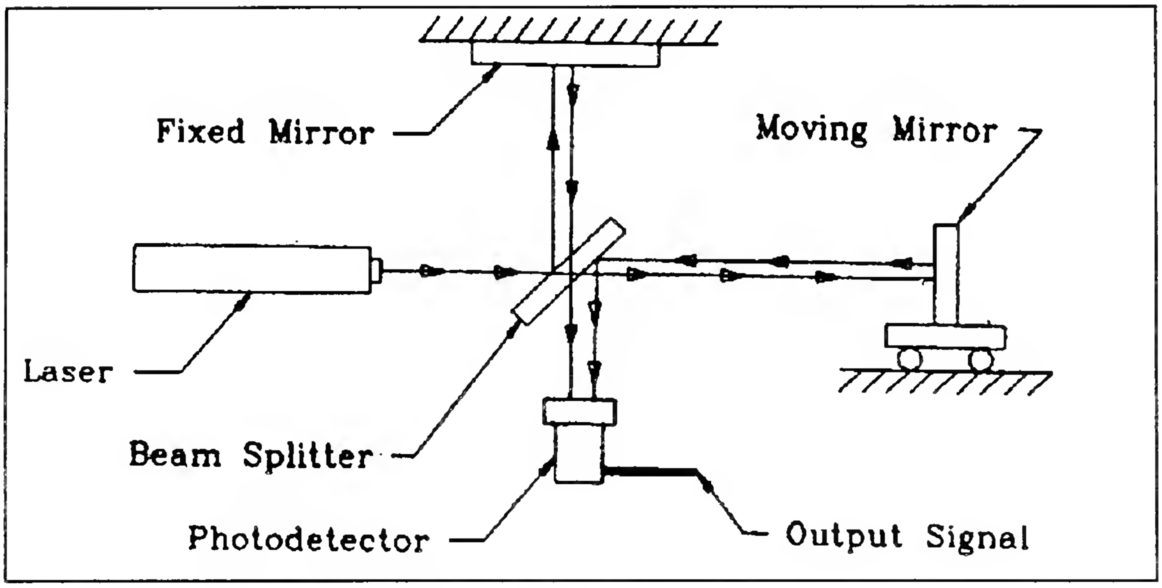


Figure 5-1: Basic layout of a laser interferometer

by laser interferometers have extremely high resolution. However, several factors should be considered for coordinate measurement. First, the interferometer readings only provide relative displacement information; there should exist one reference system in which relative displacements are measured. Second, the direction of the incident beam must be determined. Third, since it is difficult to position the mirror center, the mirror center offset, which is the distance from the mirror center to the intersection point of the incident beam with the mirror surface, should be properly compensated. However, the accuracy of a coordinate measuring machine based on a laser tracking system is primarily determined by the geometry of the tracking mirror system and its control. The major error sources are gimbal axis misalignment and mirror center offset.

There are two primary reasons for not using the laser interferometer as the first choice to measure the object coordinates. The first is that it is difficult to establish a reference position. Since the interferometer collects the information of the reflected laser beam, the measurements have to start again if the light beam is interrupted at any time. The second reason is that the interferometer is expensive. A typical commercial system with necessary optics costs more than $30,000. This cost does not include the calibration sub-system.

Camera System

Camera systems have been widely utilized to capture scenes and perform image processing with various applications. In order to apply the systems in 3-D coordinate measurements, the concept of stereo vision is applied. Mapping a 3-D scene onto an image plane is a many-to-one transformation. That is, an image point does not uniquely determine the location of a corresponding world point. The stereo vision technique could retrieve the missing depth information which provides 3-D coordinate information.

Stereo vision involves obtaining two separate imaging views of an object of interest. In Fig. 5-2, the distance between the centers of two lenses is the baseline (B). Two image points (x1,y1) and (x2,y2) are given to calculate the world coordinate (X,Y,Z) of the corresponding point. If the cameras are identical and perfectly aligned, the 3-D information could be retrieved from the stereo vision system.

A typical vision system consists of a lens and a light sensitive array. Light from the image is then focused on the array by the lens. The light sensitive array consists of a large number of discrete cells that sense the frequency of light which strikes them. These cells are referred to as pixels. The frequency information from each pixel is digitized and a number is assigned to the sensed frequency. Using the information from each cell, the image could be reconstructed and analyzed. For experimental purposes,

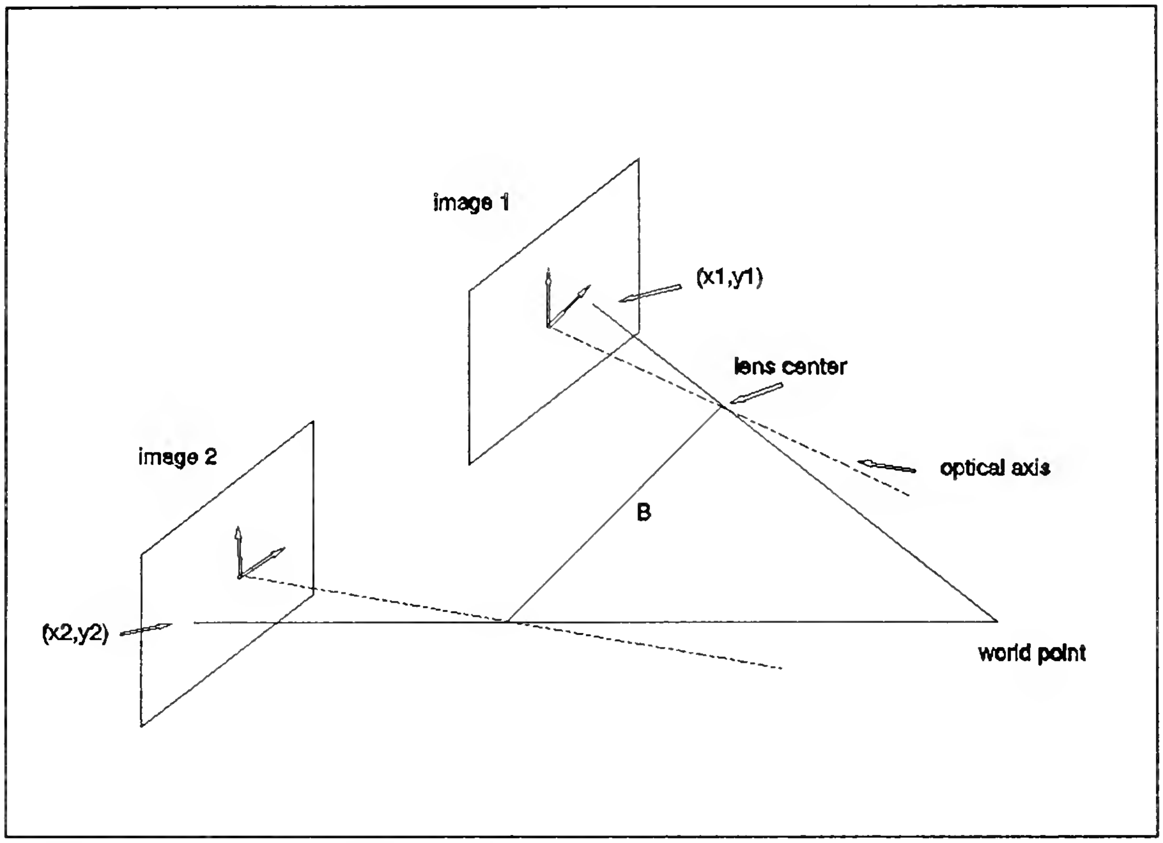


Figure 5-2: Model of the stereo imaging process

a program has been written to verify the accuracy of the stereo vision concept of the camera system from the film dosimetry setup. This program utilizes the functions imbedded in the film dosimetry project [Dub93] to execute the basic image grabbing operations. From those acquired images, a set of dots are analyzed to calculate the target location from the theory which was developed by Siddon [Sid87] to identify the BRW™ coordinates in plain radiographs. This camera system uses a Data Translation® DT2851 high resolution frame grabber board and a DT2858 frame processor board to capture and process images. A lucite plate is made as an calibration device in order to calculate the accuracy and resolution of the imaging system. It has four dots which are separated at 6 cm intervals and one dot on the cross hair of the diagonal lines. In order to examine the concept of stereo vision, a calibration experiment of the image resolution was performed as shown in Fig. 5-3. First, the lucite plate was positioned on top of the light box and those dots were easily captured from the frame grabber into the image plane. Second, in order to calibrate this camera system with respect to a specified physical dimension, a calibration factor which represents the actual dimension of each pixel (millimeter per pixel) is entered. This process helps to define the image resolution according to the user's selection. Third, a Sobel image processing filtering technique [Gon87] was performed to accurately identify the geometrical centers of the dots. The external video input signal is supplied by a CIDTEC® CID2710 Charge Injected Device (CID) camera. The CID2710 sensor chip consists of a high resolution matrix with 776 horizontal and 512 vertical pixel elements. The active area is approximately 60 mm^2. A 512x480 pixel image is stored in the buffer memory on the image processing board, and displayed on the monitor.

The center point of the calibration plate is calculated with respect to the four corner points. The error magnitude of the X and Y coordinates of the center point is an integer of the pixel resolution. Therefore, the accuracy of coordinates from those fiducial points depends on the pixel resolution. The resolution in the angio localization procedure of the UF stereotactic radiosurgery system is about 0.0254 mm. If the camera system had higher resolution than that of the existing angio localization procedure, the stereo vision concept will be available for the determination of coordinates of the intracranial target.

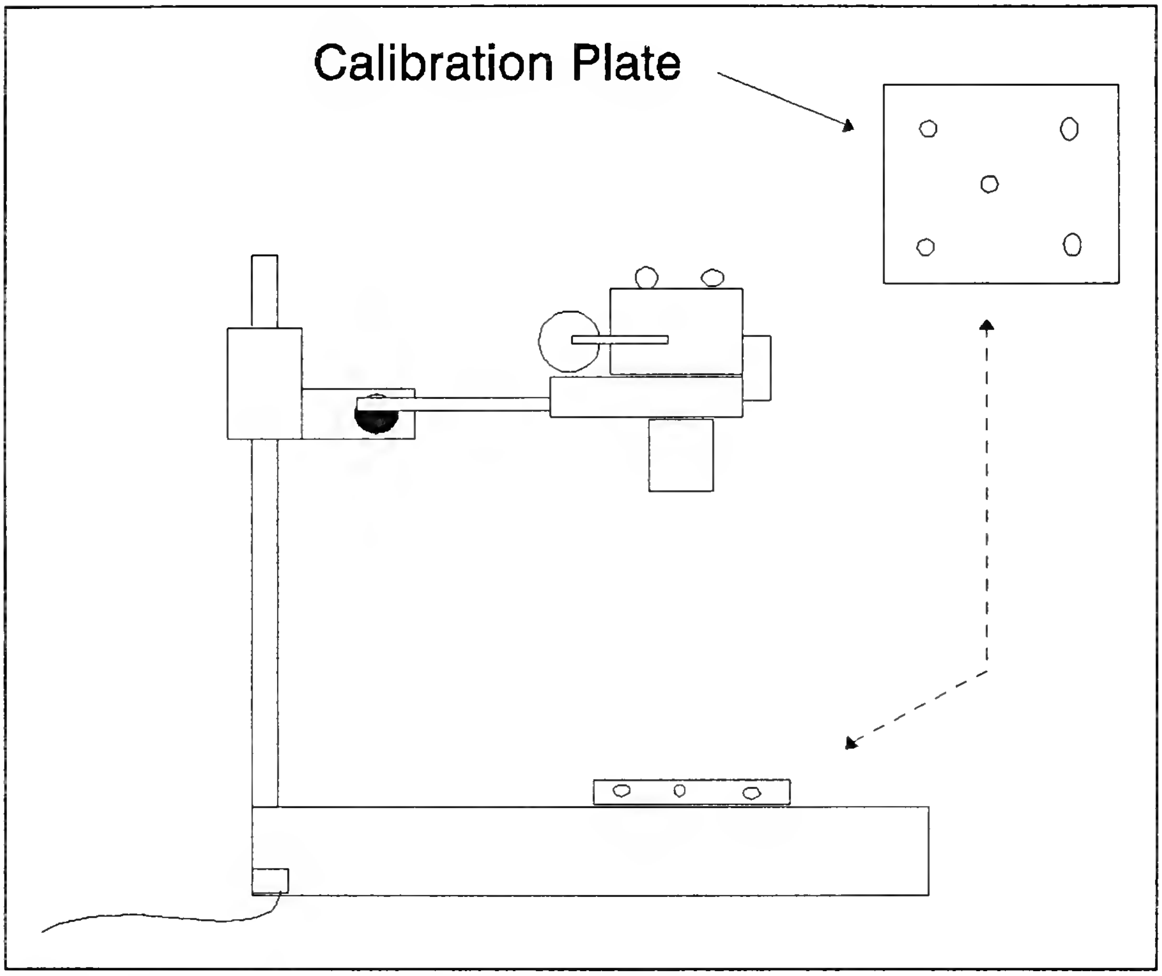


Figure 5-3: Film dosimetry system setup (camera, light box, and stand). A calibration plate is designed and analyzed for the system resolution.

Table 5-1: Pixel accuracy estimation of different calibration factors

Calibration factor (mm/pixel)	Pixel accuracy of x direction	Pixel accuracy of y direction
x_cal* = 0.005 mm y_cal* = 0.005 mm	0.297776 mm	0.233384 mm

x_cal = 0.05 mm
y_cal = 0.05 mm
	0.297019 mm	0.233392 mm

x_cal = 0.5 mm
y_cal = 0.5 mm
	0.296985 mm	0.232635 mm

*: where x_cal and y_cal represent the user defined pixel resolution.

After analyzing the images, the Table 5-1 shows the result of pixel accuracy of the camera system. The resolution determines the positions of the fiducial markers. Therefore, they will affect the solution of the determination of target coordinates.

From Table 5-1 a noticeable characteristic is observed that the accuracy of each pixel actually is determined by the CID sensor chip where the image was formed. User defined pixel resolution has no effect on the camera system. For a different calibration distance of the lucite plate, the pixel accuracy of the image is fixed with any changes of calibration factors. Therefore, the stereo vision method for the experimental purpose is limited by the resolution of the image chip unless we purchase high accuracy photosensor chip, for example, 1024x1024 array instead of 512x512 array. The resolution of the array plays an important role in measuring the 3-D coordinates. For example, assume that an array consists of 512 rows and 512 columns. If the lens system is focused on a 152.4 cm (5 ft) by 152.4 cm (5 ft) workspace, each pixel would be averaging the light from 0.3 cm by 0.3 cm square. If a resolution of 0.05 mm is required for a given identification procedure and the vision system to be used has an 512 by 512 CID array, the measurement volume will be limited to no more than a few centimeters. Therefore, it would be impractical using this passive imaging system to obtain the 3-D coordinates from the fiducial markers on the images. Only the active imaging method (i.e. with an active light or laser source at the measuring position) can accurately locate the target position if the camera system were selected. The reason is that for the passive camera method the image quality is responsible for the determination of the 3-D coordinate of the selected points. However, extra image correlation and processing efforts are always necessary to extract the coordinate information. Uncertainty of the definition for matching points on the image and localization errors might occur while performing the image correlation procedure.

The stereo vision concept has also been verified by Fitzgerald et al. [Fit75] by using two stereo shifted radiographs instead of using cameras. X-ray radiographs theoretically carry infinite resolution. The shifted radiograph technique is using a fiducial jig allows the shift parameters to be derived from information contained in the films. By translating the X-ray linearly, the depth information of the object can be calculated. However, there are potential errors when small shifts are used. This method explains another possibility of calculation error if the two cameras are not separated far enough. Besides the limited resolution of camera sensors, the geometrical structure of camera set-ups also play an important role in 3-D coordinate calculations.

Several commercial companies such as Pixsys™, Inc. and Northern Digital™, Inc. have successfully applied the stereo vision concept with an active light source to manufacture 3-D optical digitizer products. In this commercial system several charge-coupled device (CCD) cameras are positioned at different directions to capture images from any angle since only two cameras might miss some of the views. Northern Digital™, Inc. manufactures a camera type of measuring system which is called OPTOTRAK/3000 [Nor93]. This system has the capability of using a photosensitive detector that is sensitive to light in the infrared range. The sensor is capable of accurately determining the location of the center of a spot of light that has been projected onto its surface. The detector is an analog rather than digital device so the resolution is theoretically infinite. The device is also highly linear, which enhances the overall accuracy of the total system.

These systems operate by the following procedures. Several infrared Light Emitting Diodes (LEDs) are attached to the points for measurement. Then the system controller switches each LED on in sequence and the light from the LED is projected through the lens onto the photodiode. The X-Y coordinates on the image plane of the detector are proportional to the X-Y location of the LEDs in the field of view. Each LED could be located in approximately 50 μsec so the system can record the trajectory of those LEDs. Therefore, those cameras can track and report the 3-D coordinate location of the LED. Those data sets can be transferred into the workstation or PC for 3-D graphics and for image correlation. According to the manufacturer's specification, they have 0.01 mm resolution at 2.5 m, with an accuracy of 0.1 mm in the X-Y direction, and 0.15 mm in depth which is perpendicular to the X-Y plane.

Pixsys™ also utilizes a array of CCD cameras which are placed above the patient table. These cameras track and report the 3-D coordinates of tiny LEDs which are built into the instruments such as stylus, forceps, biopsy needles or ultrasound transducers. The 3-D coordinates are reported up to 100 times per second. If the CT or MR data sets are preloaded to the computer, then the corresponding images from these data sets can be displayed from the locations of LEDs which shows the instruments' location relative to anatomical landmarks. However, camera resolution is 0.3 mm in one cubic meter measuring volume and 0.6 mm in two cubic meter measuring volume, which is worse than the Northern Digital camera system.

Articulated Arm System

Articulated arm systems for 3-D coordinate measurements have been widely used in aerodynamics or machinery. These devices are manufactured to ensure precise motion along the desired axes and they are instrumented to determine the joint displacement and orientation to a high degree of accuracy. The basic structure of this type of mechanical system consists of joints and arms; a set of potentiometers or encoders are also necessary to read out the spatial position and orientation. Such devices typically have a resolution of 0.025 mm and a total accuracy on the order of 0.125 mm [Moo91].

Articulated type of coordinate measurement systems are typically designed for inspection of parts and assemblies. In the clinical uses, such coordinate measurement systems are quite cumbersome because the heavy weight and geometrical structures often interfere with either the diagnostic or therapy procedures. Watanabe et al. [Wat87] utilized a 3-D digitizer for CT stereotaxic surgery. This digitizer has multiple aluminum arms and joints which consist of high resolution potentiometers. Analog signals from the potentiometer are linear with respect to the angle of the joint. The 3-D coordinates of the tip of the arm are calculated from the joint angles and each segment length. Another group in Germany developed a robot type of system in house to perform computer-assisted surgery. Adams et al. [Ada90] developed an electromechanical robot for 3-D coordinate measurements. This digitizer can interact with CT image data sets and perform a real-time surgery. The system has an accuracy of ± 1 mm which is typical in robotic standards. All the articulated type of systems have the following characteristics:

(1) Heavy and fixed mechanical systems, cumbersome in clinical settings;

(2) Lengthy time for measuring a large number of contour points;

A company named Shooting Star Technology™ had developed a inexpensive 6-D (translation plus rotation) tracking localization system ADL-1™. However, the resolution is at the order of 0.635 mm for displacement and 0.3 degrees for orientation, which depends on the position in work volume. Apparently it is not suitable for high precision clinical uses.

Frequency Matching System

Instead of using mechanical arms and joints, Polhemus™, Inc. developed a real-time, 3-D data generation and manipulation system for motion tracking. This system utilizes a radiofrequency transmitter and receiver for emitting and sensing the electromagnetic signals. From the geometrical relationship, the controller automatically sends out and records the 3-D information. Because the system can track objects in real time, if the sensor is attached to a stylus, the 3-D coordinate information can express the object outlines which are being traced.

The accuracy of this digitizing system ranges from 3.3 mm (0.13 inches) RMS (Root Mean Square) from 101.6 mm (4 inches) to 381 mm (15 inches) source-to-sensor separation. As the separation increases, the positional accuracy degrades linearly. Although this system has reasonable accuracy for measuring 3-D coordinates, the drawback is that large metallic objects, such as desks or cabinets, located near the source or the sensor adversely affect the performance of the system. If the patient is positioned under the treatment machine, the metallic parts and the electromagnetic pulses will interfere with the proper reading of the 3-D coordinates. Therefore, this system is excluded for our application.

Ultrasonic System

The ultrasonic system for determination of 3-D coordinates of defined points is also clinically available in neurosurgical operations. The operational principles of the ultrasonic navigation system are similar to those of the frequency matching system. There are several commercial products which perform the range-finding function as well as calculate the spatial coordinates of the selected points. Science Accessories Corp. (Southport, CT) developed the Model GP-8-3D™ ultrasonic range-finding device as a 3-D navigation system. This range finder is computer controlled and calculates distances between sound emitters and microphones by measuring the transit time of an acoustic pulse traveling from a sound emitter to the microphones. The ultrasonic system was examined with respect to all the possible errors, either systematically and randomly, to quantify the possibility of clinical use. Friets et al. [Fri89b] studied the different errors in this ultrasonic system. The range finder error is 0.5 mm $\pm$ 0.2 mm with a 0.2 mm standard deviation. The average determination error of actual microphone separation is 0.2 mm $\pm$ 0.1 mm with 0.1 mm standard deviation. Transformation of the emitter signals to the image plane introduces up to 4.3 mm $\pm$ 2.3 mm with a standard deviation for the orientation point (the furthest point from the focal plane of the system). The minimum error of the system while locating a point from the real operating environment in CT space is 2.2 mm $\pm$ 1.0 mm with a standard deviation of 0.5 mm. They claimed that the phantom registration can achieve as low as 2 mm and 3 mm display error of the contours (matrix transformation error). Environmental parameter such as temperature can also influence the accuracy of the ultrasonic system because the medium for carrying the signals is air. Change of the medium characteristics influences the response curve of the ultrasonic system. Clinically, the ultrasonic system has been proven as one of the candidates to perform real time image registration during operations.

Conclusion

After a careful selection procedure, the Northern Digital™ camera has been selected as the 3-D coordinate measuring device for the following reasons:

(1) High resolution at certain long distances. For example, it has a resolution of 0.01 mm at 2.5 m distance.

(2) Pre-calibrated 3-D coordinate system which eliminates user calibration process. Once the camera is properly calibrated, the results are stable indefinitely.

(3) Sampling rate of 3,500 markers per second and 3-D data sets available in real time for viewing during data collection. From using the real time digitized probe, contour line data and the selected anatomical points can be obtained by an experienced user within a short time, perhaps a few minutes. Therefore, the calculation of the transformation matrix between the patient scanning image data and the data set from the patient treatment setup location is also possible in the fast and accurate manner.

(4) Interface with PC, which makes an easy operating front end in the clinical environment. The required information can be retrieved and computed by simple command line language which needs less human interference.

CHAPTER 6
THE FITTING OF 3-D REFERENCE DATA SETS

Basic Principles

While performing motion analysis of a rigid body, the estimation of the 3-D motion parameters which are described as rotation and translation is important. This analysis can be very useful in many applications such as scene analysis, motion prediction and trajectory planning in the defense industry. For fractionated stereotactic radiotherapy the patient needs to be set up at the correct treatment position during the multiple treatment courses. Solving the refixation problem requires the matching of 3-D surface and anatomical data points on the rigid object at two different times. Therefore, the patient relocalization procedures are similar to that of the estimation of motion parameters. Determining the position and orientation of the head for each treatment fraction is an important aspect of patient relocalization. The mathematical background for solving the refixation parameters of position and orientation is indeed the process of fitting two sets of 3-D data points to the patient's contour. Since the patient's head can be considered as a rigid object, the 3-D points on the patient's head could provide necessary information for the calculation of repositioning parameters.

When the patient experiences the initial stereotactic radiotherapy treatment, the CT/MR scanning data sets and the corresponding coordinates of contour points on the patient's face provide the geometrical relation of tumor location. However, during fractionated radiotherapy, the well calibrated 3-D localizer provides coordinate information for those fiducial points on the patient's surface. How to correlate these two sets of data in order to reposition the tumor location to match the machine isocenter without using any invasive relocalization method becomes a key issue to our research. The methodology of the contour match solution is to find the best fit via translation and rotation of an optical 3-D data set with a corresponding 3-D CT/MR data set.

In order to determine the patient relocalization parameters, the following mathematical problem is encountered. The initial diagnostic scans of the patient consist of 3-D geometrical relationship between the tumor and patient surface structures. This is a complete standard data set for treatment planning. If patient comes in for the fractionated treatment in another time frame, the interior anatomy can not be observed and only the surface information can be retrieved. Rigid body transformation indicated that if the patient surface information of two treatment time schemes have the best fit and then the tumor should be at the correct treatment coordinates. It is equivalent to find the best correlation parameters for two sets of 3-D data. Therefore, assume that we are given two 3-D point sets $p_i=[x_i,y_i,z_i]^t$, $p'_i=[x'_i,y'_i,z'_i]^t$, $i=1,2,3,\ldots,N$, (here p_i and p'_i are considered as 3x1 column matrices.) The two sets of points are related by:

$$p_i' = Rp_i + T + N_i \qquad (6.1)$$

where R is 3x3 rotation matrix, T is a translation vector (3x1 column matrix), and N_i is a noise factor. The rotation matrix and translation vectors are answers for best correlation of the data sets but with highly nonlinear functions, which means difficult to solve analytically (no exact answers are available).

Huang et al. [Hua86] has shown that rotation and translation can be decoupled by a centroid coincidence theorem. This theorem uses the differentiating technique of Lagrangian multipliers and provides groundwork for the determination of rotation and translation parameters (three from rotation and three from translation) separately. Assume that R and T generate the least-square best fit of the following equation:

$$\xi^2=\sum_{i=1}^{n} \|p_i^{\prime}-(Rp_i+T)\|^2 \tag{6.2}$$

Therefore by minimizing the above least-square term the centroids of $\{p'_i\}$ and $\{Rp_i+T\}$ can be proved to be coincident, i.e., the centroid after rotation and translation is the same as that of the least-square solution of proper rotation and translation. Mathematically, the centroid coincidence theorem can be shown as follows:

$$p^{\prime} = p^{\prime\prime} \tag{6.3}$$

$$p^{\prime} \triangleq \frac{1}{n} \sum_{i=1}^{n} p_i^{\prime} \tag{6.4}$$

$$p^{\prime\prime} \triangleq \frac{1}{n} \sum_{i=1}^{n} p_i^{\prime\prime} \triangleq \hat{R}p + \hat{T} \tag{6.5}$$

where n is the total number of contour data points.

This centroid coincidence theorem implies that we can simplify the original least-square problem by translating the point sets first, and the rotation parameters can be solved accordingly. From the above theorem, we clearly understand that if we want to find the solutions of rotation and translation parameters, the following logical procedures should be performed:

(1) Simplify the original least-square problem by translating the point set $\{p_i\}$ by the following equation:

$$T_c \triangleq p' - p \tag{6.6}$$

where T_c is the translation vector. Notice that the T_c is not the translation vector which gives the best least-squared fit answer. It is only the estimation of the centroid difference between two 3-D data sets. The accurate translation vector should be calculated after the rotational manipulation, which indicates the rotational sequence is before the translation. The order is important because that matrix multiplication is not commutative.

(2) From the centroid coincidence theorem, the coordinates of two centroid points are calculated. Then the 3-D points of the two are translated by subtracting the corresponding values from the new origin.

(3) Performing the fitting method to acquire the rotation parameters, i.e., find the R vector to minimize the following equation:

$$\xi^2 = \sum_{i=1}^{n} \| Q_i' - R Q_i \|^2 \tag{6.7}$$

where

$$Q_i \triangleq p_i - p \tag{6.8}$$

$$Q_i' \triangleq p_i' - p' \tag{6.9}$$

(4) The translation vector is calculated by utilizing the following relation:

$$\hat{T}=p'-\hat{R}p \tag{6.10}$$

(5) The six variables which relate rotation and translation vectors are the best fitting of the least-square function of two 3-D point data sets.

The rationale for using the centroid coincidence theorem is to decouple the translation and rotation operations into two separate parts. Using this technique can reduce the parameter searching space from 6-D (rotation and translation) to 3-D (rotation only). The calculation time and complexity of the problem is also reduced. In other words, rotational angles are actually the primary unknowns in our mathematical approach since the translation vectors can be calculated in a fast and accurate way after least-squared rotational manipulation.

In order to solve the least-square minimization problem several methods are developed. The selection of different methods totally depends upon personal choice. The detailed description of different approaches are introduced. The six fitting parameters can be calculated to realign patient treatment positions by following the centroid decoupling methodology and the results are compared.

The Iterative Method

Huang et al. [Hua86] utilized an exhaustive iteration method to search for the converged answer. He decomposed the rotation matrix into three successive rotations around z-axis, x-axis, and y-axis, respectively. The rotation matrix can be described as:

$$R=R_y(\psi)\,R_x(\phi)\,R_z(\theta) \tag{6.11}$$

where ψ is the rotation angle on y-axis

ϕ is the rotation angle on x-axis

θ is the rotation angle on z-axis

The least square term can be rewritten as:

$$\xi^2(\psi,\phi,\theta)=\sum_{i=1}^{N}\|q_i^{\prime}-R_y(\psi)R_x(\phi)R_z(\theta)q_i\|^2 \tag{6.12}$$

This approach minimizes least-square fitting errors with respect to one variable at a time while restraining the other two axes with fixed angles. The method transforms a 3-D problem to a 2-D problem and the error will eventually converge to the minimum.

The iterative method needs initial guesses of three rotation angles for the function value. Using the projection technique onto a 2-D plane where the angles are reassumed, a new least-square value can be calculated with a new rotation angle. The diagrammatic concept is shown in Fig. 6-1. By revolving through three different orthogonal axes, the new least-square value can be calculated and minimized at each plane. When the function value approaches or is less than some preset threshold, the process is terminated. The three final rotation angle for each axis is the solution. According to the simulation work which was performed by Huang et al., it was discovered that the function value of the least-square term does have several saddle points. They claimed that even though the saddle points exist, the algorithm will find the minimum value, since the minimum value was calculated in 2-D for each step.

The Singular Value Decomposition (SVD) Algorithm

The SVD algorithm for matrix operation is a powerful technique for dealing with sets of equations or matrices that are either singular or else numerically very close to

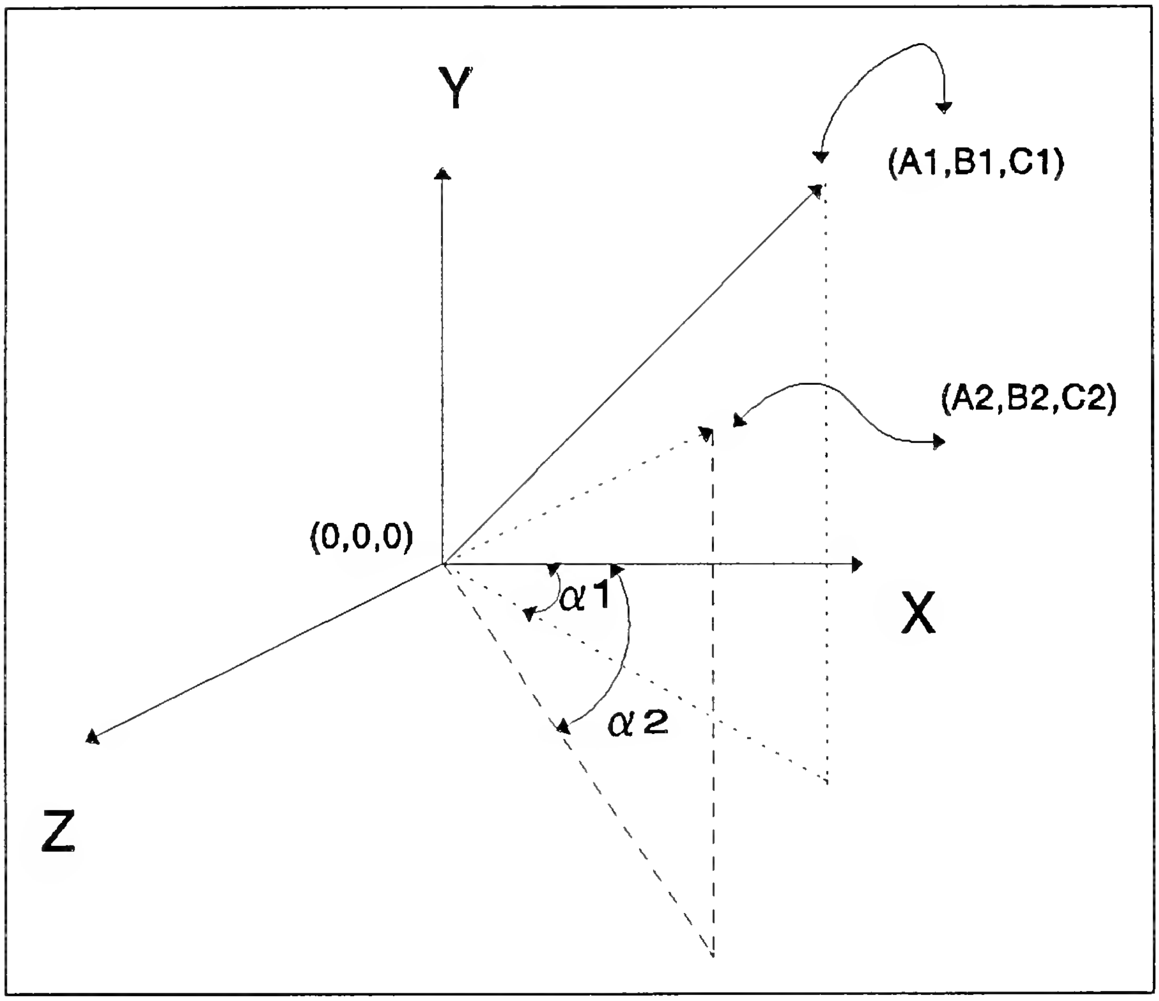


Figure 6-1: Using projection plane to calculate the iteration angle for each rotation

singular. It is also the method of choice for solving most linear least-squares problems where the number of equations is larger than the number of unknowns. This method starts with a set of simultaneous equations that can be reformatted into a simple matrix equation:

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b} \tag{6.13}$$

where A is a matrix, b and x are vectors. Equation (6.13) defines A as a linear mapping matrix from the vector space x to the vector space b. For example, consider the problem of fitting a polynomial $p(x)=a_1+a_2x+a_3x^2+\ldots+a_Nx^{N-1}$ of degree N-1 through M data points (x_i,y_i), $i=1,\ldots,M$. The coefficients $a_1,\ldots,a_N$ must satisfy the M linear equations to deliver a minimized least-squared solution. Equation (6.14) shows a matrix multiplication form for solving the linear equations.

Inverting a matrix and getting parameters for fitting a set of linear equations are

$$\begin{vmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{N-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & x_M & x_M^2 & \ldots & x_M^{N-1} \end{vmatrix} \begin{vmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ a_N \end{vmatrix} = \begin{vmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ y_M \end{vmatrix} \tag{6.14}$$

straightforward by using standard numerical methods such as continuous iteration, Gaussian elimination or QR decomposition [Ken89]. However, if these linear equations are close to linear dependence or have been classified as rank deficit, inverting a matrix becomes a very tedious and unstable procedure. The above algorithms sometimes fail to calculate the right matrix and the answers are easily diverged. However, the SVD numerical analysis method explicitly constructs orthonormal bases for the matrix and decomposes the matrix into proper mathematical format. Therefore, inverting the decomposed mathematical matrix form becomes much simpler because these terms are well defined and no divergence will occur even if the matrix function is close to singular.

A simplified representation for the linear transformation A defined by the m x n matrix A by introducing two orthonormal bases can be easily computed. A decomposed matrix multiplication of A can be found at the cost of finding a great deal of information about the eigenvalue system related to A. The arbitrary orthonormal set of vectors $[u_1,\ldots,u_n]$ which is used for the domain space n x n are generated while the other arbitrary orthonormal set of vectors $[v_1,\ldots,v_m]$ is used in the domain space m x m, i.e.,

$$\boldsymbol{U}=[u_1,\ldots,u_n] \qquad (6.15)$$

$$\boldsymbol{V}=[v_1,\ldots,v_m] \qquad (6.16)$$

Then any m x n matrix A whose number of rows m is larger than or equal to the number of columns n (points are larger than unknowns), can be written as a product:

$$\boldsymbol{A}=\boldsymbol{U}^T\cdot\Sigma\cdot\boldsymbol{V} \qquad (6.17)$$

where A is an (m x n) matrix

U and V are (n x n) and (m x m) unitary orthonormal matrices

$\Sigma = \Sigma(w_{ij})$ satisfies $w_{ij} = 0$ if $i \neq j$

The decomposition of matrix A can always be done, no matter how singular the matrix is. A proof that any matrix could have SVD matrices is beyond the scope of this paper, interested readers should refer to the reference on matrix multiplication [Gol89]. The SVD algorithm organizes the following representation for the inverse matrix of A. The inverse matrix A^{-1} is immediately solved as follows:

$$\boldsymbol{A}^{-1}=\boldsymbol{V}\cdot\Sigma^{-1}\cdot\boldsymbol{U}^T \qquad (6.18)$$

Thus x and b can be related as:

$$\boldsymbol{x}=\boldsymbol{V}\cdot\Sigma^{-1}\cdot(\boldsymbol{U}^T\cdot\boldsymbol{b}) \qquad (6.19)$$

If b and x have some systematic errors involved or more equations than unknowns, the problem turns out to be that there is no linear mapping between b and x. The closest answer we encounter finds the least-squares solution. The calculation of least-square minimization, which relates the two sets of data points with a best estimated rotation matrix, is based on the SVD algorithm. To apply the SVD algorithm in solving the rotation matrix in the subspace of these data points, the following method approaches the goal in generating a unique and minimum matrix for rotation estimation.

The use of the SVD algorithm to estimate the least-squared parameters is based upon the following discussion. Suppose that $A \in \mathrm{IR}^{m \times n}$ is a data matrix obtained by performing a certain set of experiments. If the same set of experiments is performed again, then a different data matrix, $B \in \mathrm{IR}^{m \times n}$, is obtained. The possibility that B can be rotated into A with the rotation matrix R is explored by solving the following problem:

$$\text{minimize } \| A-RB \|_F \qquad \text{subject to } R^T R = I_p$$

This is a typical application for solving the least-squares problem. If the least-squared answer is achieved, the rotation matrix provides the best match between two sets of data points. Note that if $R \in \mathrm{IR}^{n \times n}$ is orthogonal, then expanding the above equation we have:

$$\begin{aligned} &\|\boldsymbol{A}-\boldsymbol{RB}\|_F^2 \\ &=\sum_{i=1}^{N} (\boldsymbol{A}-\boldsymbol{RB})^T(\boldsymbol{A}-\boldsymbol{RB}) \\ &=\sum_{i=1}^{N} (\boldsymbol{A}^T\boldsymbol{A}+\boldsymbol{B}^T\boldsymbol{B}-2\boldsymbol{A}^T\boldsymbol{RB}) \\ &=Trace(\boldsymbol{A}^T\boldsymbol{A})+Trace(\boldsymbol{B}^T\boldsymbol{B})-2Trace(\boldsymbol{A}^T\boldsymbol{RB}) \end{aligned} \tag{6.20}$$

where Trace is defined as

and a_{ii} is the diagonal element of the matrix A.

$$Trace(A) = \sum_{i=1}^{N} a_{ii} \qquad (6.21)$$

Therefore, the above equation for solving least-squares minimization is equivalent to the problem of maximizing the Trace(A^TRB). Maximizing R can be found by calculating the SVD of A^TB. Indeed, if $U^T(A^TB)V = \Sigma = diag(a_1,\ldots,a_p)$ is the SVD of the matrix and we can define the orthogonal matrix Z by $Z = V^TRU$, then:

$$\begin{aligned} Trace(RA^TB) &= Trace(RU\Sigma V^T) = Trace(Z\Sigma) \\ &= \sum_{i=1}^{p} z_{ii}a_i \leq \sum_{i=1}^{p} a_i \end{aligned} \qquad (6.22)$$

In order to prove that equation (6.22) is valid, we have to prove the following lemma first:

Lemma: For any positive definite matrix AA^T, and any orthonormal matrix B, the following equation exists.

$$Trace(\boldsymbol{BAA}^T) \leq Trace(\boldsymbol{AA}^T) \qquad (6.23)$$

Proof of Lemma: Let a_i be the ith column of A. Then:

$$Trace(BAA^T) = Trace(A^TBA) = \sum_{i} a_i^T(Ba_i) \qquad (6.24)$$

From the Schwarz inequality, the following equation is valid,

$$a_i^T(Ba_i) \leq \sqrt{(a_i^Ta_i)\,(a_i^TB^TBa_i)} = a_i^Ta_i \qquad (6.25)$$

Hence,

$$Trace(BAA^T) \leq \sum_i a_i^T a_i = Trace(AA^T) \qquad (6.26)$$

Clearly, the upper bound is attained by setting the $R = VU^T$ for $Z = I_p$. Therefore, the procedures to maximize the trace of equation (6.23) can be explained as the following algorithm.

This algorithm to solve the over-estimated system can be summarized in several logical steps. The following explanation about the SVD algorithm can be applied to solve the rotation matrix for least-squared matching in two 3-D data sets.

Given A and B in $\mathrm{IR}^{m\times n}$, the following algorithm finds an orthogonal $R \in \mathrm{IR}^{n\times n}$ such that $\| A-RB \|_F$ is minimum.

(1) Calculate $C = A^TB$

(2) Compute the SVD $U\Sigma V^T = C$, then $U^TCV = \Sigma$. Save U and V.

(3) Obtain $R = VU^T$

(4) Then we have:

$$RA^TB = VU^TU\Sigma V^T = V\Sigma V^T \qquad (6.27)$$

We know that the trace of RA^TB has the largest value. Then the VU^T is the rotation matrix we want and is the orthogonal polar factor of A^TB.

From the above algorithm, the matrix R which would minimize $\| A-RB \|_F$ is calculated. This matrix delivers the closest solution for matching two 3-D data sets A and B. The matrix R could be further transformed into three orthogonal Euler transformations which are as follows:

where c represents cosine function and s represents sine function. Respectively, α, β and

$$Q=\begin{vmatrix} c\alpha c\beta & s\alpha c\gamma+c\alpha s\beta s\gamma & s\alpha s\gamma-c\alpha s\beta c\gamma \\ -s\alpha c\beta & c\alpha c\gamma-s\alpha s\beta s\gamma & c\alpha s\gamma+s\alpha s\beta c\gamma \\ s\beta & -c\beta s\gamma & c\beta c\gamma \end{vmatrix} \tag{6.28}$$

γ are three Euler angles for Yaw, Pitch and Roll angles in the orthogonal coordinate system. With proper matrix manipulation, the three rotation Euler angles can be calculated from the relationship contained inside the matrix function. If we assume the resulting matrix as another form such as the equation (6.29), the angles are easily calculated inside the matrix.

$$Q=\begin{vmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{vmatrix} \tag{6.29}$$

where α = Atan2(-r21,r11)

β = Atan2(r31,sqrt($r11^2+r21^2$)

γ = Atan2(-r32,r33)

when β = 90° => $\alpha = 0$, γ = Atan2(r12,r22)

β = -90° => $\alpha = 0$, γ = -Atan2(r12,r22)

If the experimental data have some types of systematic errors (such as digitization errors), this rotation matrix will apparently give the best estimation of the rotation angles.

Arun et al. [Aru87] developed a non-iterative algorithm which involves the SVD of a 3x3 matrix to solve the least-square fitting of two 3-D point sets. The purpose was to compare different algorithms in matching two 3-D data points and the conclusion was that the SVD algorithm is the most stable and the fastest among iterative methods.

Pelizzari et al. [Pel91] has applied the SVD algorithm to successfully relocalize the patient's treatment position. A system of aligning two sets of 3-D patient contour data was reported. The surface fitting method determines the transformation between two coordinate systems by finding the transformation which best matches models of the same surface as defined in each of the coordinate systems.

Fragments of the source codes obtained from Pellizzari and Chen which were written in C on VAX/780™ have been modified and adapted into the University of Florida PC system. This program provides the least-squares solution of the 3-D point matching problem. This source code has been extensively modified because of system differences and the specification parameters of six degrees fitting. This modified software package will be considered as the standard protocol for comparing the calculated results which are obtained from the methods of optimization techniques.

The Vector Analysis of Robotic Method

A potentially useful method for 3-D point fitting is robotic vector analysis. Since six-degree motion parameters are generally an important topic in robotic analysis, a geometrical approach is performed to solve the matching problem. Similar to robotic manipulators, the mechanisms for solving translation vectors and rotation angles are dominated by highly non-linear trigonometric equations. The approach here is to accomplish the solutions by relating the translational and rotational mechanisms directly to the trigonometric formulation. This procedure provides deeper understanding of geometrical representation of implicit correspondence between two data sets using homogeneous matrices.

The vector analysis starts with initiation of a certain type of mechanism (please refer to the following Fig. 6-2). From the designed mechanism, we can analyze the geometrical relationship of different joints. The mechanical setups of each arm and joint are confined in advance to limit the movements. Thus the translation and rotation parameters can be calculated with the kinematic equations by both forward and inverse analyses. The manipulator study is to detail the algorithm which models the six degree-of-freedom serial manipulator into a single closed loop spatial mechanism of mobility one. The algorithm is explained and organized as follows:

(1) First determine the location of three fiducial marks with respect to a fixed coordinate system. Determine relationship between the "head" coordinate system and the fixed coordinate system, $({}^{F}T_{H})$.

(2) Perform the forward analysis to determine the relationship between the fixed coordinate system and the head ring system. Let us define the fixed coordinate system as $({}^{F}T_{6})$ and invert the coordinate as $({}^{6}T_{F})$.

(3) Determine the relationship between the fixed coordinate system and the head coordinate system by $({}^{6}T_{H}) = ({}^{6}T_{F})({}^{F}T_{H})$.

(4) Determine the required position and orientation of the head ring system so that the reference points are in the necessary position.

(5) Then perform the reverse analysis to determine the joint values to move reference points to the required position. If properly defined for all the joint values, such as limitation of each slide length and each rotation angle, it is possible only one solution will fall within allowable joint limits.

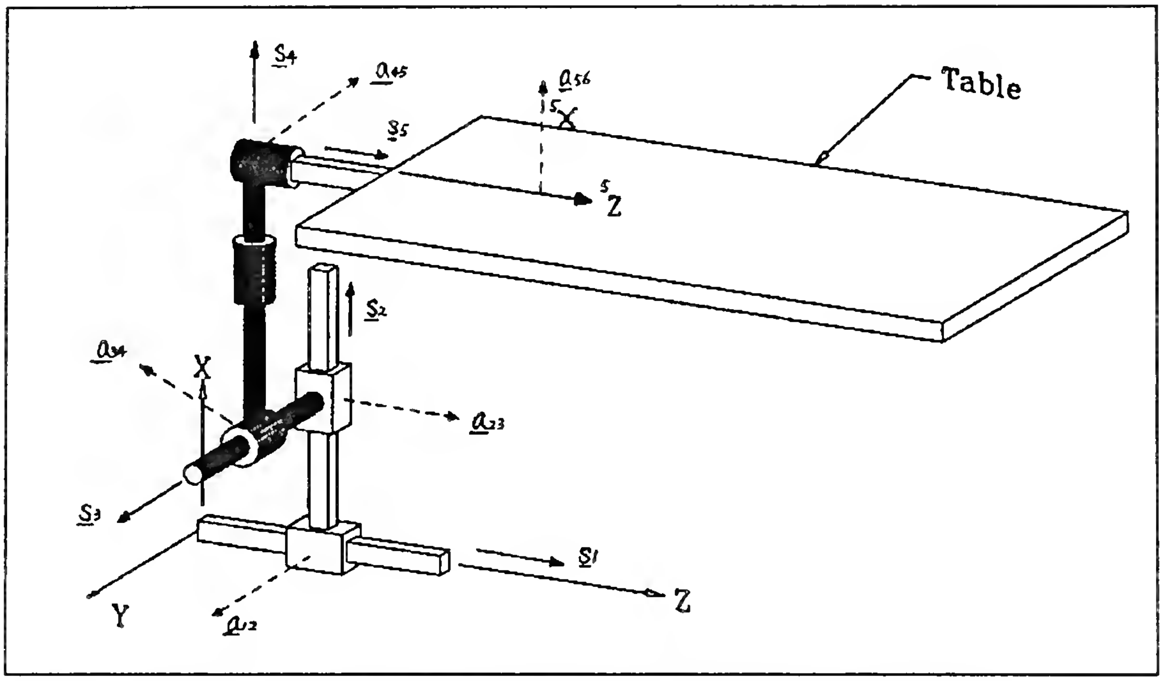


Figure 6-2: Kinematic model of the apparatus in solving the translational and rotational parameters

However, the above approach of robotic vector analysis requires exact mapping of two data sets. If the noise vector in equation (6.1) becomes larger than a certain level, this algorithm will not solve the correct coordinates of the 3-D data points. Therefore, error analysis is important while performing coordinate transformation. In other words, if this algorithm is applied, the optimization procedure should be incorporated into the process to obtain the least-squared errors of the two data sets. For demonstration purposes, the program runs on a Silicon Graphics™ workstation with the geometric mechanism displayed.

The Optimization Methods

Utilizing optimization techniques to solve the highly non-linear least-squared problem is a promising approach. The purpose of this approach is to apply optimization algorithms and to find the optimum parameters for automatic rotation and translation for patient relocalization. Numerical optimization techniques have been widely used in industry to improve production efficiency [Van84]. The optimization method provides improved accuracy, less computation time, and logical design procedures.

Since we are dealing with a multidimensional optimization problem (i.e., three rotation and three translation parameters), the simplicity of one-dimensional optimization has been lost. The optimization criteria is to minimize the object function, which is the least-squared difference between two 3-D patient contour and anatomical point data sets. The numerical optimization techniques are heavily dependent on personal taste. A method which does not require any derivatives of the design function was chosen, which could save a lot of computation time. This direct approach depends upon the converging criterion of the least-square function for two 3-D data points.

Three different optimization techniques are performed to calculate and compare the rotation and translation parameters for repositioning the patient's treatment position. Therefore, the irregular shape of a tumor for fractionated treatment is closely correlated during different treatment courses.

Hooke and Jeeves Pattern Search Method

The Hooke and Jeeves pattern search technique is a climbing method that does not require the use of derivatives. The algorithm has ridge following properties and is based on the premise that any set of design moves that have been successful during early experiments will be worth trying again. The method is based on the assumption of a simple unimodality of the object function value and is used to find the minimum of a multivariable, unconstrained function of the form:

$$Object\ function = F(x_1, x_2, \ldots x_n) \qquad (6.30)$$

The simple logic diagram for this method can be seen in Fig. 6-3 and the detailed mathematical chart is shown as Fig. 6-4. The algorithm proceeds as follows. First, a base point in the feasible design space is chosen together with exploration step sizes. For example, three Euler angles are assumed zero as the initial guessing values. Next, an exploration is performed at a given increment along each of the independent-variable directions following the logic shown in Fig. 6-5 and Fig. 6-6.

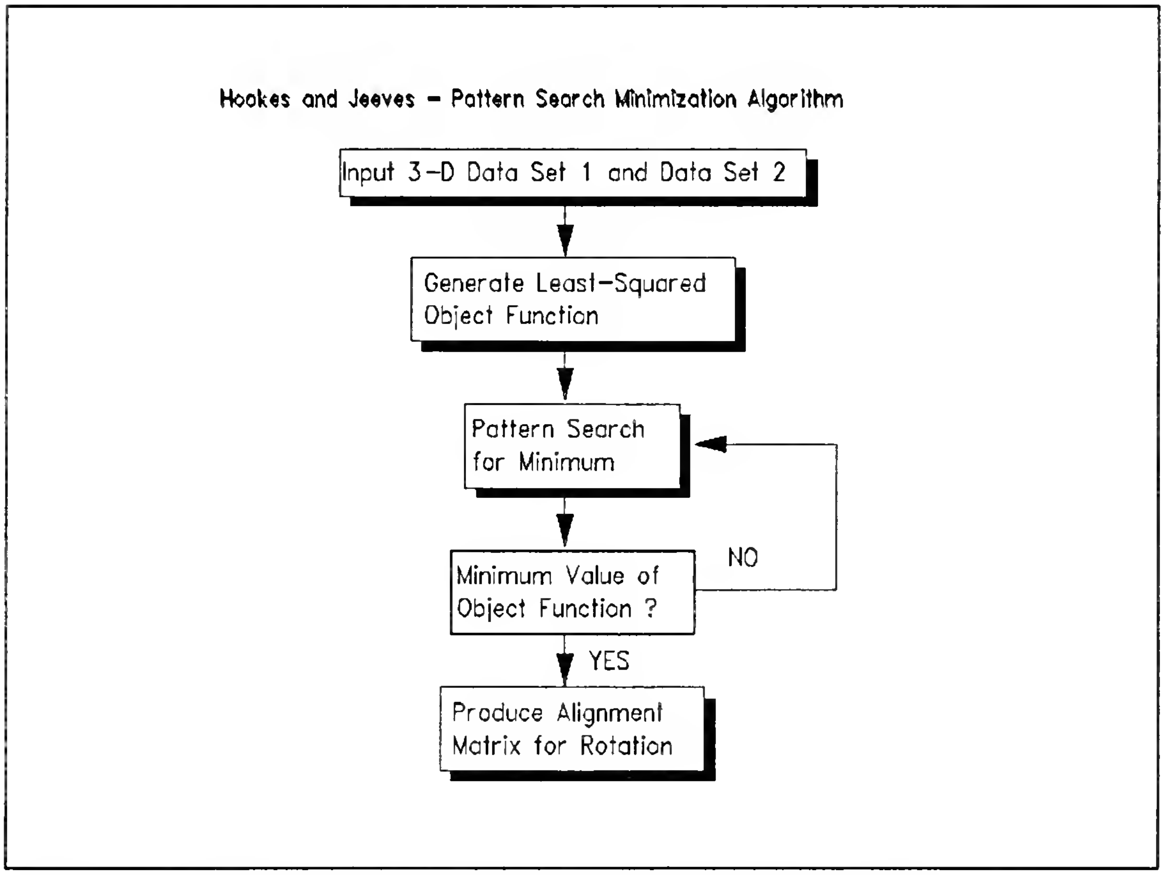

Hookes and Jeeves - Pattern Search Minimization Algorithm
Input 3-D Data Set 1 and Data Set 2
Generate Least-Squared Object Function
Pattern Search for Minimum
Minimum Value of Object Function ?
NO
YES
Produce Alignment Matrix for Rotation

Figure 6-3: Logic diagram of Hooke and Jeeves optimization algorithm

Whenever a functional improvement (improvement of the object function value) is obtained, a new temporary base point is established. Once this exploration is complete, a new base point is established and a new "pattern move" takes place. This pattern move consists of an exploration along a line between the new base point and the previous base point. Mathematically, this exploration is defined as:

$$B_{i,0}^{(k+1)} = B_i^{(k+1)} + a\,(B_i^{(k+1)} - B_i^{k}) \qquad (6.31)$$

where $B_{i,0}^{(k+1)}$ becomes a new temporary base point or "head." In this expression i is the variable index, k is the linear stage index, and a is an acceleration factor that is greater than or equal to 1.0. Once the new temporary base point has been found, an exploration about this point is instituted to see if a better base point can be found. If the answer is "yes", the pattern proceeds with this improved base. When the step size has been decreased below a predetermined value and still no substantial change in the merit value of object function can be achieved, the procedure terminates. Because of its simplicity to program, this method is one of the most popular techniques for computer implementation.

The merit function is set to calculate the minimum errors between two data sets. In calculating the rotation parameters, the least-squared minimum value of the 3-D coordinates between two translated data points has been chosen as the object function. The object function is expressed as the following format:

$$\begin{aligned} \mathit{Object\ Function} &= F(x_i, y_i, z_i) \\ &= \sum_{i=1}^{n} \sqrt{(x_{1,i}-x_{2,i})^2 + (y_{1,i}-y_{2,i})^2 + (z_{1,i}-z_{2,i})^2} \end{aligned} \qquad (6.32)$$

where n is the total number of contour points.

If the Hooke and Jeeves optimization method can search for the minimum value of the above object function, the spatial planes which are formed by two sets of 3-D data points should be exactly aligned. That means the two sets of 3-D points are matched by proper translational and rotational operations. The calculated parameters (Δx, Δy, Δz, α, β, γ) can perform the six degree-of-freedom coordinate transformation in the working space.

The Downhill Simplex Method in Three Dimensions

The Downhill Simplex method is due to Nelder and Mead [Nel65]. A simplex is an n-dimensional, closed geometric figure in space that has linear line edges which intersects at (n+1) vertices. For example, in two dimensions this figure would be a triangle. In three dimensions it is a tetrahedron. Search schemes are based on the simplex utilizing the function values of each vertex. The basic move in this method is to find a reflection point of the Simplex and generate a new vertex point and a new simplex. The choice of reflection direction and the new vertex points relies on the location of the worst point in the simplex. The new reflected point is called the *complement* of the worst point [Pre92]. If the algorithm oscillates back and forth instead of going toward to the extremum, the second worst point should be used to locate the complement point. The Downhill Simplex method only requires function evaluation rather than derivative evaluation.

The simplex points are manipulated by reflection, contraction, and expansion. This simplex method adapts itself to the local landscape, elongating down long inclined planes, changing direction on encountering a valley at an angle, and contracting in the

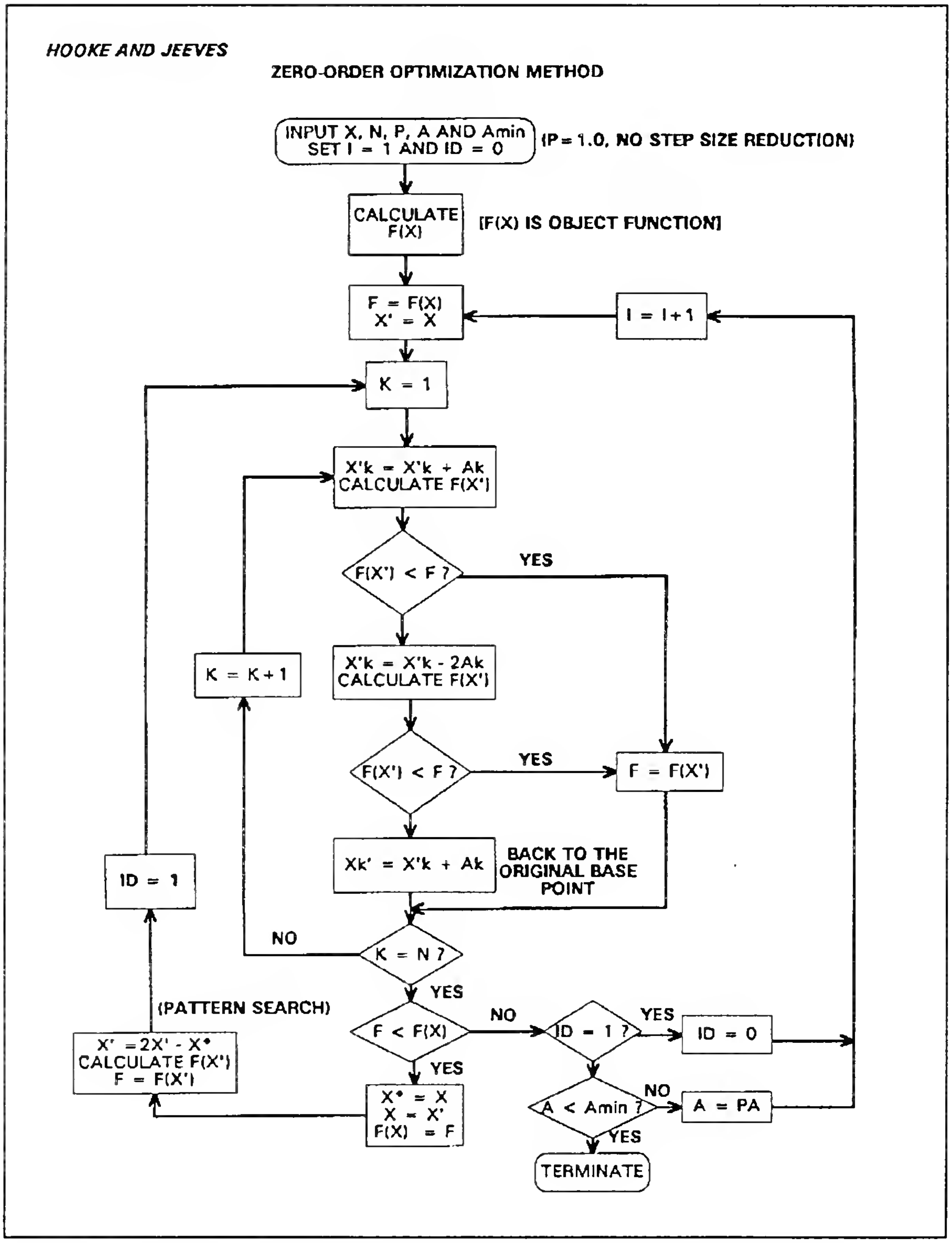

HOOKE AND JEEVES
ZERO-ORDER OPTIMIZATION METHOD
INPUT X, N, P, A AND Amin SET I = 1 AND ID = 0
(P = 1.0, NO STEP SIZE REDUCTION)
CALCULATE F(X)
(F(X) IS OBJECT FUNCTION)
F = F(X) X' = X
I = I+1
K = 1
X'k = X'k + Ak CALCULATE F(X')
F(X') < F ?
YES
K = K+1
X'k = X'k - 2Ak CALCULATE F(X')
F(X') < F ?
YES
F = F(X')
ID = 1
Xk' = X'k + Ak
BACK TO THE ORIGINAL BASE POINT
NO
K = N ?
YES
(PATTERN SEARCH)
F < F(X)
NO
ID = 1 ?
YES
ID = 0
X' = 2X' - X* CALCULATE F(X') F = F(X')
YES
X* = X X = X' F(X) = F
A < Amin ?
NO
A = PA
YES
TERMINATE

Figure 6-4: Hooke and Jeeves optimization process (X is input parameters, N is dimension of searching space, P is factor of step size decrement, and A is the increment on each dimension which can be justified)

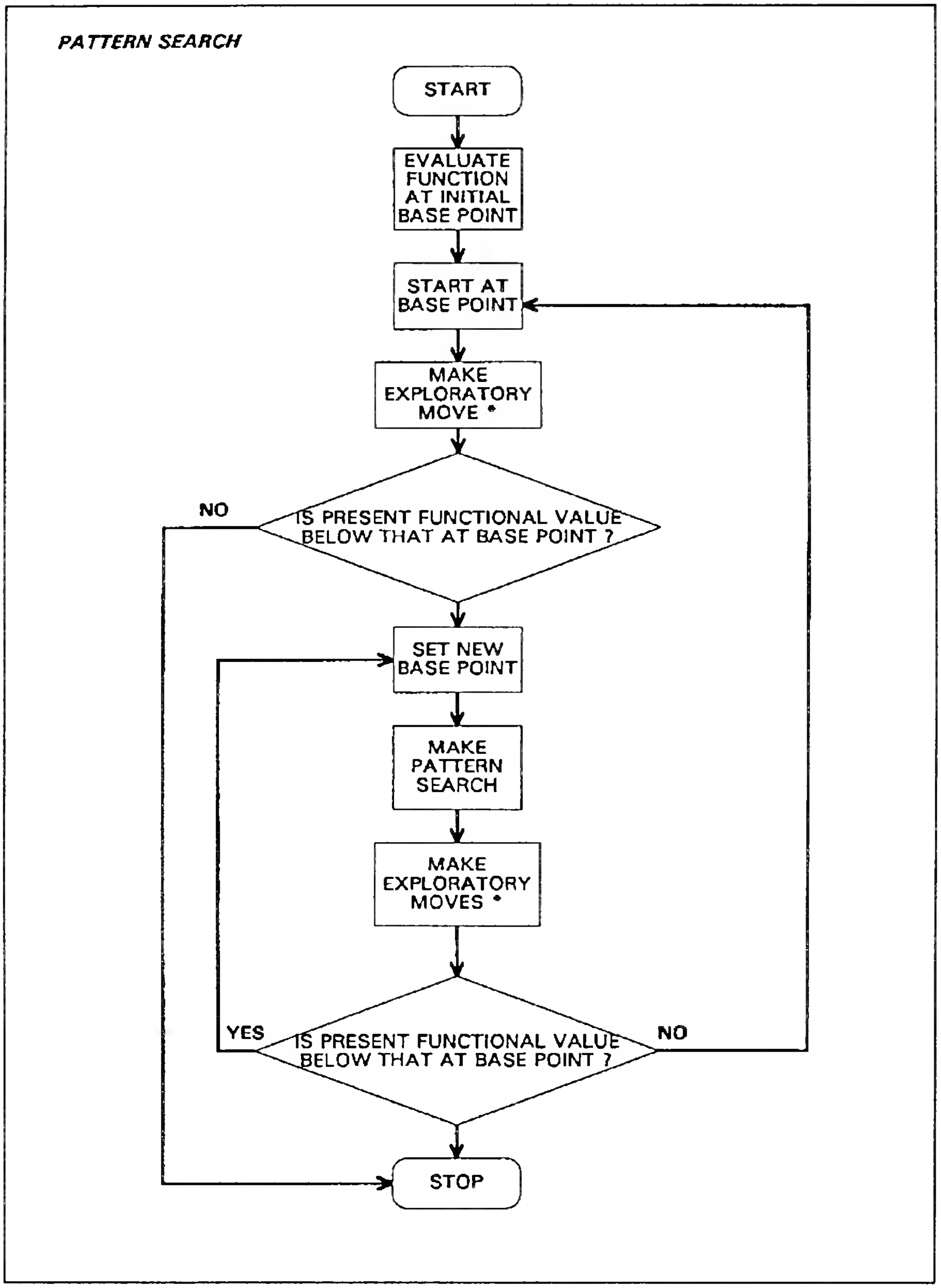

PATTERN SEARCH
START
EVALUATE FUNCTION AT INITIAL BASE POINT
START AT BASE POINT
MAKE EXPLORATORY MOVE *
IS PRESENT FUNCTIONAL VALUE BELOW THAT AT BASE POINT ?
NO
SET NEW BASE POINT
MAKE PATTERN SEARCH
MAKE EXPLORATORY MOVES *
IS PRESENT FUNCTIONAL VALUE BELOW THAT AT BASE POINT ?
YES
NO
STOP

Figure 6-5: Pattern search in the Hooke and Jeeves optimization process

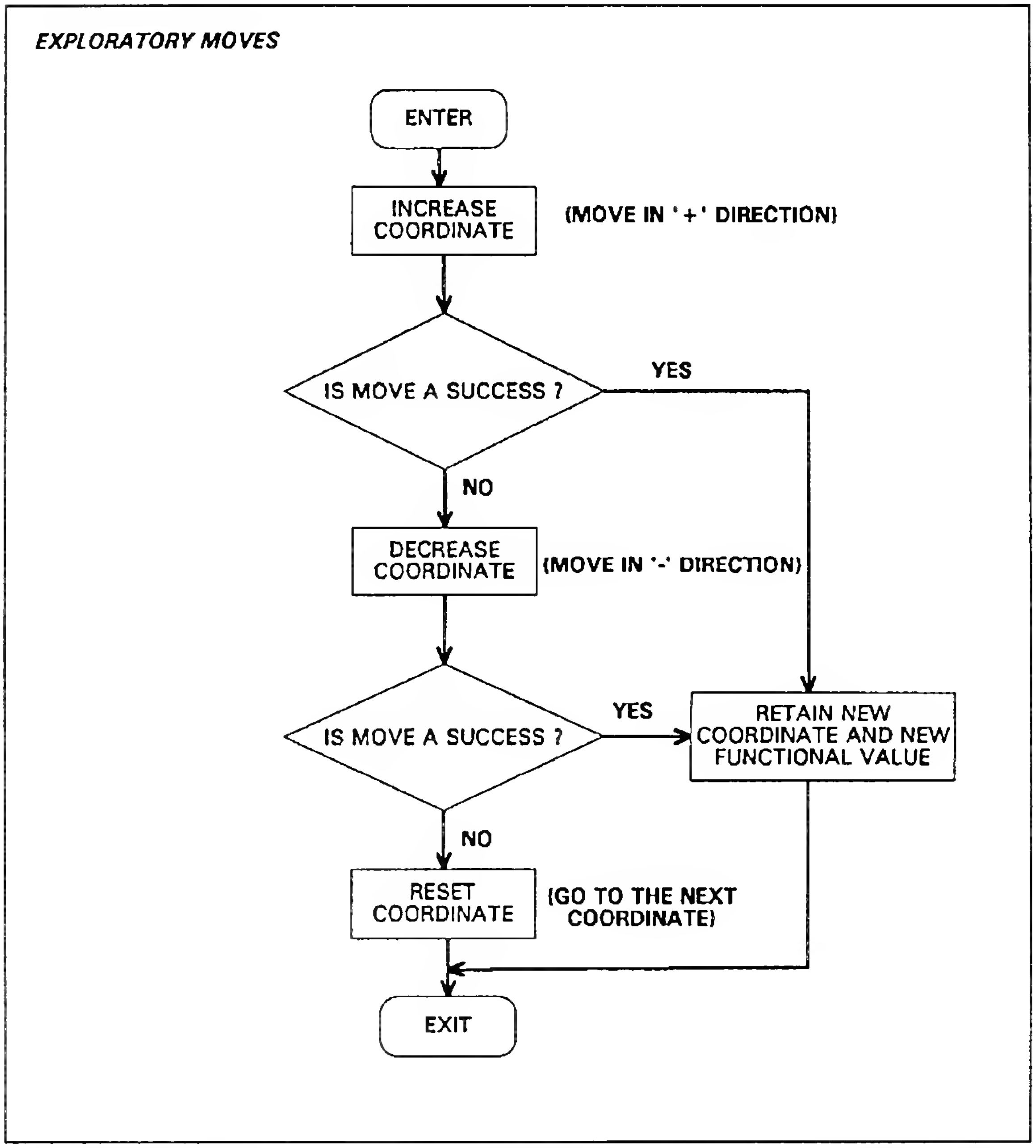


Figure 6-6: Exploratory move in the pattern search approach of the Hooke and Jeeves optimization process

neighborhood of a minimum [Nel65]. By repeating the reflection, contraction or expansion procedures, this action contracts the simplex towards to lowest point, and will eventually bring all points to the minimum value. The simple diagram which describes the optimization method is shown in Fig. 6-7.

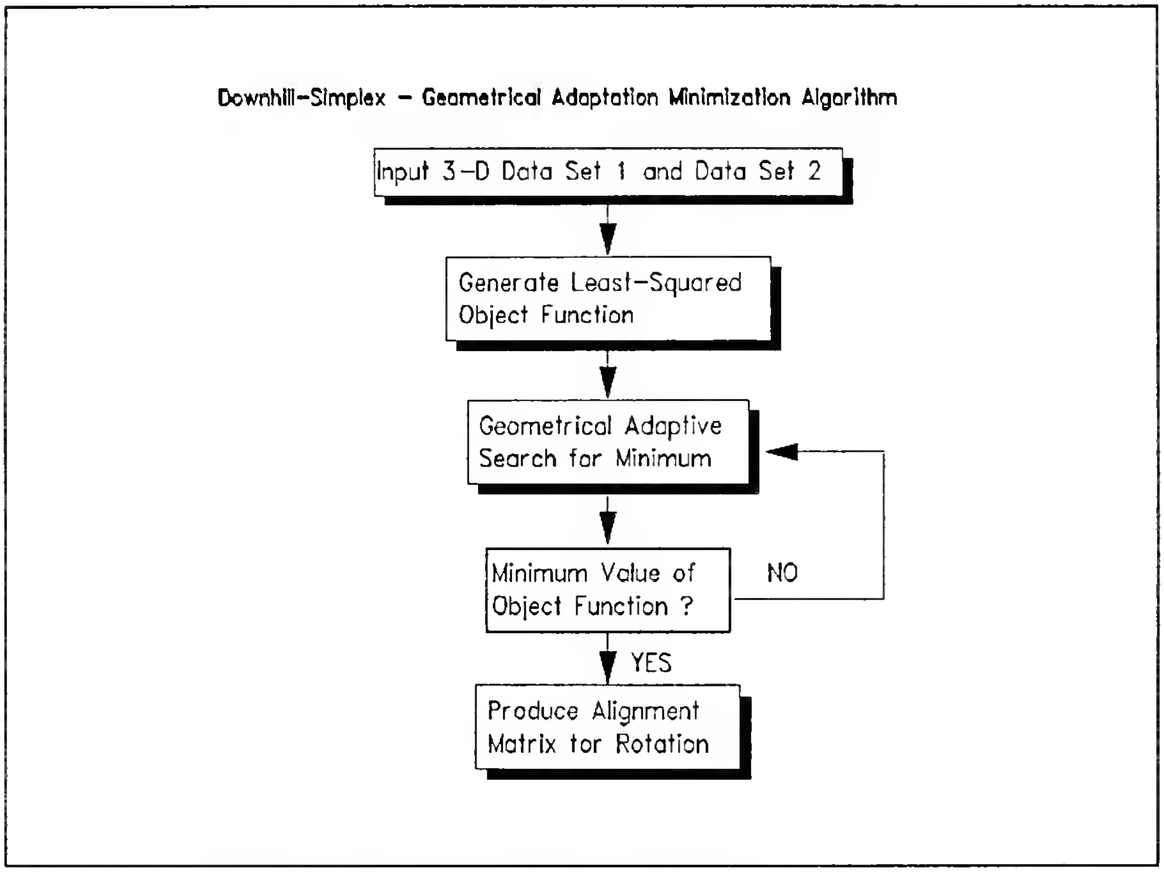

Downhill-Simplex - Geometrical Adaptation Minimization Algorithm
Input 3-D Data Set 1 and Data Set 2
Generate Least-Squared Object Function
Geometrical Adaptive Search for Minimum
Minimum Value of Object Function ?
NO
YES
Produce Alignment Matrix for Rotation

Figure 6-7: The diagram of Downhill Simplex optimization algorithm

There are two difficult situations in this type of optimization technique. First, the answers might be the local minimum points instead of the global minimum points. This difficulty has been found in using the Simplex method on a four-dimensional surface having a long, curved, valley with extremely steep sides; along the valley bottom the function varies considerably compared with the required accuracy for the minimum function value [Nel65]. In other words, the calculated results may be trapped by the saddle points. Secondly, computer execution time might not be economical if too many function variables are involved. To solve the rotation matrix and translation vector for patient relocalization in fractionated stereotactic radiotherapy, the patient position will be closely repositioned by the laser before fine adjustment. It means that the motion parameters we seek are fairly close to the global minimum points. Chance of being trapped at a local minimum point will be decreased to an acceptable level. Besides, we only have six design variables for repositioning the patient's head, therefore the PC is a suitable tool for the optimization calculation. The detailed mathematical description of Downhill Simplex method in a flow chart format can be seen in Fig. 6-8.

The Simulated Annealing Optimization Method in Three Dimensions

The object function in the determination of the rotation parameters (three Euler angles) consists of nonlinear variables and may have multiple minima. The condition of nonlinear variables excludes certain minimization methods such as linear or quadratic programming, which need a specific object function format. The condition of multiple minima is the limitation of some quick optimization algorithms because they might yield a solution at the nearest local minimum value. A parameter estimation method that is resistant to becoming trapped by local extrema is named Simulated Annealing. The usefulness of the Simulated Annealing method comes from the possibility of accepting a "uphill" move in the parameter space. That is, instead of moving towards to the minimum value, the searching process sometimes jumps up to the higher function value, which is against the search trend. This method prevents the function value from being

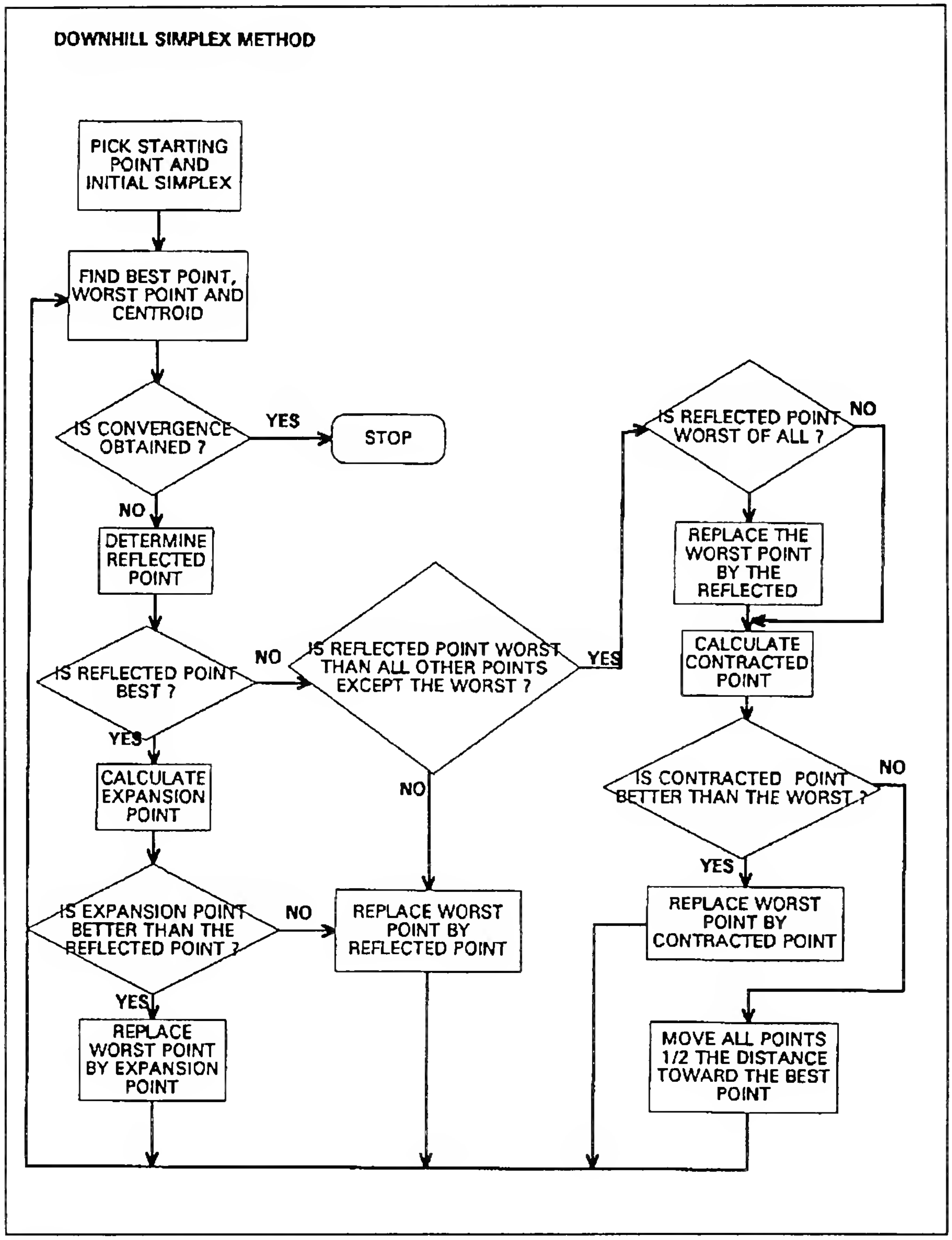

DOWNHILL SIMPLEX METHOD
PICK STARTING POINT AND INITIAL SIMPLEX
FIND BEST POINT, WORST POINT AND CENTROID
IS CONVERGENCE OBTAINED ?
YES
STOP
NO
DETERMINE REFLECTED POINT
IS REFLECTED POINT BEST ?
NO
YES
CALCULATE EXPANSION POINT
IS EXPANSION POINT BETTER THAN THE REFLECTED POINT ?
NO
YES
REPLACE WORST POINT BY EXPANSION POINT
IS REFLECTED POINT WORST THAN ALL OTHER POINTS EXCEPT THE WORST ?
YES
NO
REPLACE WORST POINT BY REFLECTED POINT
IS REFLECTED POINT WORST OF ALL ?
NO
REPLACE THE WORST POINT BY THE REFLECTED
CALCULATE CONTRACTED POINT
IS CONTRACTED POINT BETTER THAN THE WORST ?
NO
YES
REPLACE WORST POINT BY CONTRACTED POINT
MOVE ALL POINTS 1/2 THE DISTANCE TOWARD THE BEST POINT

Figure 6-8: The Downhill Simplex optimization algorithm for the calculation of 6-D fitting parameters

trapped in the local minimum and provides the necessary search to the global minimum. The optimization process for this geometrical minimization problem can be observed in

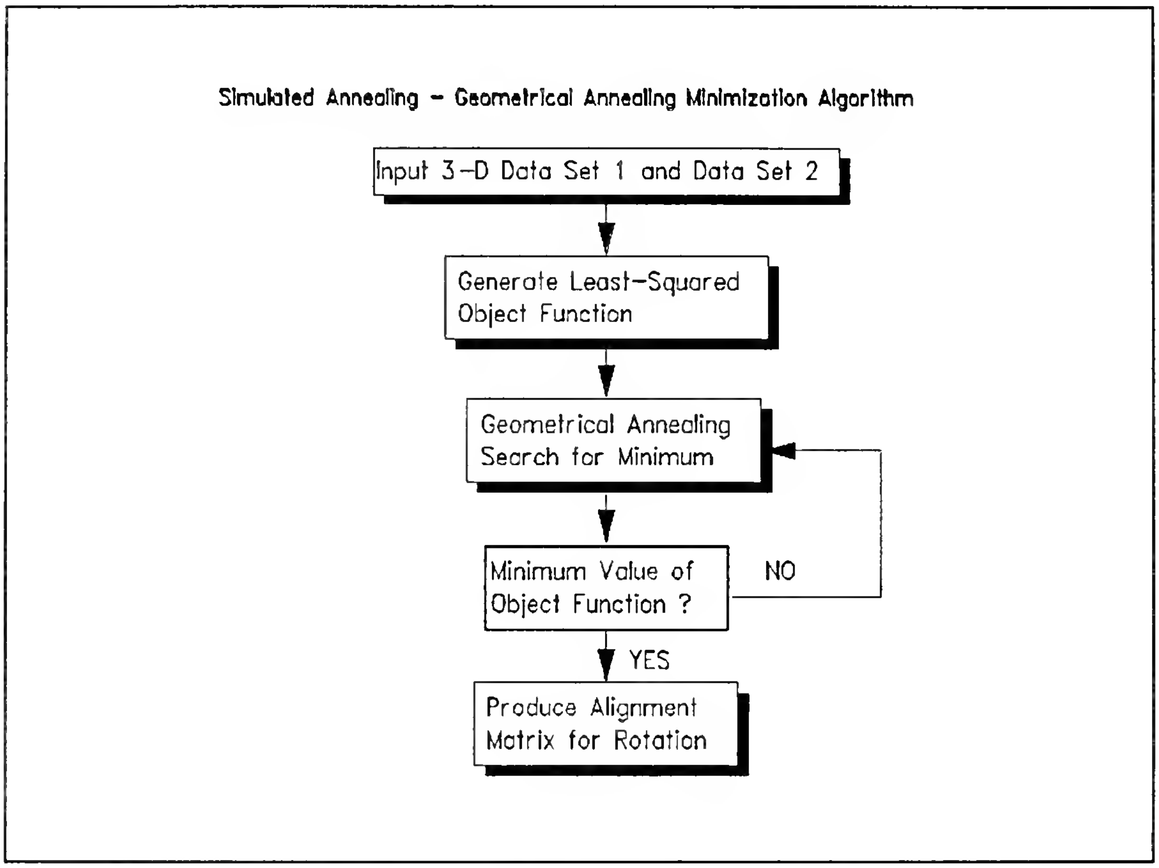

Simulated Annealing – Geometrical Annealing Minimization Algorithm
Input 3–D Data Set 1 and Data Set 2
Generate Least–Squared Object Function
Geometrical Annealing Search for Minimum
Minimum Value of Object Function ?
NO
YES
Produce Alignment Matrix for Rotation

Figure 6-9: The diagram of Simulated Annealing optimization algorithm

the Fig. 6-9. The basic idea of the Simulated Annealing method is also applicable in solving the three-dimensional least-squared minimization problems for the calculation of rotational parameters. With continuous three dimensional spaces, this algorithm will ideally find the global minimum in the presence of many local minima. The four elements required by the Metropolis procedure in Simulated Annealing algorithm are as follows [Met53, Boh86]:

(1) The value of the object function, in our case, which is the least-squared value of the

difference from two 3-D data sets.

(2) The system state at the point x, which is the vector at that specified state.

(3) The control parameter T, which is similar to the temperature, with an annealing schedule by which it is gradually reduced. T is also an important factor to specify the "cooling schedule" or "cooling curve".

(4) A generator of random changes, taking a random step from x to $x+\Delta x$.

Simulated Annealing consists of three basic steps sequentially to constitute one iteration of the procedure [Met53, Kir83]:

(1) A random generator function, which starts with the current configuration or initial value of object function, generates a new function value. The magnitude of all the changes to the variables of each iteration is called step size, which can be characterized inside the program according to the user's preferences.

(2) An acceptance function which accepts or rejects the new configuration based on the changes ΔV of the object function. If the object function value decreases, the new configuration is always accepted; otherwise it is randomly accepted with a probability analogous to the Boltzmann distribution $\exp(-\Delta V/T)$, where T, the temperature, controls the degree of "uphill climbing" of the object function.

(3) An updated function which determines the step size and temperature is used in the next iteration procedure.

There is an easy way to perform the Simulated Annealing procedure. It is possible to combine simulated annealing with a version of Downhill Simplex searching that allows upper and lower bounds on all parameters [Pre92]. The Simulated Annealing procedure can come close to the true global minimum in parameter space, while the simplex search, now begun near the global minimum, can search for a true global minimum without fear of falling into the local minimum value. The implementation procedure of the Metropolis scheme (cooling curve) is by adding a positive, logarithmically distributed random number (it will always give out the positive term), proportional to the annealing temperature T, to the stored function value associated with every vertex of the simplex geometry. Then subtract a similar random variable from the function value of every new point that is tried to replace the old vertex. Like the standard Metropolis process, this method always accepts a true downhill step, but sometimes accepts the uphill one. The selection of downhill or uphill steps is to provide the random process and the local minimum trap can be avoided.

At a finite value of T, the simplex expands to a scale that approximates the size of the region that can be reached at this temperature, and then executes a stochastic, tumbling Brownian motion within that region, sampling new, approximately random, points as it does so. If the temperature is reduced sufficiently slowly, it becomes highly likely that the simplex will shrink into the region containing the lowest relative minimum encountered.

The Simulated Annealing algorithm is a fairly new method for industrial or even medical uses. However, this method provides several extremely attractive features, which supercede other optimization algorithms.

First, it is not easy fooled by the "quenching" procedure which is used for almost all the other optimization methods. Provided that sufficient general reconfigurations are given, it wanders around freely among those local minima of depth less than T. Therefore, as T decreases, the number of qualifying for frequent visits of the minimum values are gradually reduced.

Second, changes that cause the greatest energy differences are shifted over when the control parameters are large. The potential advantage of this method is that the step size adjusts itself automatically, expanding or contracting to a scale that approximates the size of the configuration space that can be reached at a given value T. This decision becomes more permanent as T gets lower. Smaller refinements are then achieved to the real solution.

Simulated Annealing of the Downhill Simplex algorithm has been applied to determine the rotation angles in the 3-D data fitting procedures. In essence, Simulated Annealing is used to provide the starting vertices for a Downhill Simplex search, thus minimizing the risk of simplex searching being trapped in a local minimum. The converging criteria is that optimization stops when the difference in the object function values of the highest and lowest configurations in the simplex is less than 0.001 [Pre92]. The Simulated Annealing algorithm provides a comparative basis for calculating the rotation parameters of the three Euler angles.

Conclusion

SVD algorithm is a standard technique to solve the least-squared over-estimated system in the numerical analysis. Solving rotation matrix is a good representation of the mathematical method for 3-D data correlation. Different optimization methods have been utilized to design the engineering products or analyze various system performance.

However, they have not been examined to solve the 3-D data correlation problems. The relationship between two patient treatment position can be evaluated and cross-calibrated through different optimization methods. Along with the SVD algorithm, systematic studies and performance evaluation are pursued. The following chapters will compare the effectiveness and convenience to correlate patient treatment position. In a clinical oriented environment, these optimization algorithms provide a very useful tool to perform patient repositioning and refixation for high precision fractionated radiotherapy treatments.

CHAPTER 7
ANALYSIS OF RELOCALIZATION ERRORS

Comparison of Algorithms

There are four different algorithms being compared in this section, namely SVD algorithm, Hooke and Jeeves algorithm, Downhill Simplex algorithm and Simulated annealing algorithm. These methods provide a solid basis for calculation of the six degrees-of-freedom fitting parameters.

In order to observe the results of these four algorithms and compare their potential clinical utilization, two studies were performed. One study used the same clinical data sets from Pelizzari [Pel89, Pel92] and compared the error terms of those algorithms. The least-squared object function values for those algorithms during optimization are also considered and plotted to demonstrate the manipulation process. The other study randomized the input data set to generate sets of stochastic outputs and performed inverse calculations for finding the best-fit parameters. A total of one hundred random outputs from the same input data were produced to perform the statistical analysis. Reverse mapping of the input and output data sets reveals the stability and feasibility of the optimization algorithms. The analysis of relocalization errors are summarized as follows :

(1) Analysis of clinical digitized data set of patient:

Two sets of clinical patient digitization data provided by Pelizzari [Pel92] were utilized in this study. The two data sets consist of independent 3-D points from a Polhemus™ digitizer for different time periods. The first data set carried 18 points of data, and the second carried 51 points by randomly selecting anatomical points of the patient surfaces. The calculation results of the error estimation are shown in Table 7-1 and Table 7-2. From the results, the optimization algorithms potentially reveal smaller error terms. Since each digitized point includes different sources of errors, then the mean error, mean square error and the root mean square error are the most appropriate forms to analyze the merit function of the 6-D fitting process.

The optimization procedures are basically the searching steps for the minimum function values. The least-squared function values of the two data sets can be approximated and minimized through the minimization algorithms. Continuous evaluation of the function values shows the 6-D fitting parameters are improved during the calculation process and the closest answers can be obtained once the pre-set tolerance value is reached.

Fig. 7-1 and Fig. 7-2 show that the Hooke and Jeeves algorithm is always searching for the better function value through each iteration. The pattern search algorithm looks for the sharpest decrease of function value and finds the best object minimization value within specific tolerance, which is the difference between two consecutive object function values during the searching process (defined as 0.0001 in the calculation algorithms). The function value is finally stabilized and reaches the minimum. This method is effective and fast if the object function is not very complicated. The 6-D fitting parameters are easily extracted from the program and show the best match for the two 3-D data sets. Observation of the graph shows that a sharp ridge occurs in the first few iterations and then the object function stabilizes for the further evaluation. This is the typical ridge characteristic of the Hooke and Jeeves algorithm. This quick converging method can bring the object function value to its minimum within a very short time and introduces no singular problem (the invert matrix does not exist) which appears often in matrix manipulation.

In order to observe object function changes in the Downhill Simplex method, the same methodology used in the Hooke and Jeeves algorithm is applied to analyze the trend in each iteration step. Fig. 7-3 and Fig. 7-4 show the function value changes during the iteration process. Fluctuation of the function values during each iteration step is observed simply because of the reflection, extrapolation and contraction procedures which occur on the simplex vertices. Geometrically, the function values will finally search for the minimum state and calculate for the best 6-D fitting parameters.

A simulated annealing process is also incorporated into the optimization algorithm to verify the calculated results, which are the best fitting parameters for the two data sets. The inverse calculation might deliver ambiguous answers. For example, instead of moving a patient to the best-fit values, the calculation might give a false local minimum that does not necessarily relocate the patient's treatment position and orientation. With the possible global searching characteristics of the annealing algorithm, the final calculation can be compared as a bench mark for the optimization algorithm. For each iteration loop a total number of 25 random iteration steps are performed with three random number generating processes. Inside the first loop of iteration, a function value is selected (not necessarily the smallest) as a starting base. Then the random process continues and compares with the previous object function value. If there were no improvement, maintain the original function value. Otherwise, the function value is replaced with the new randomized value. This process will continue until a pre-set threshold or tolerance is reached and the whole procedure is terminated. The random operation provides the perturbation which brings out the global minimum object function value. Fig. 7-5 to Fig. 7-8 demonstrate the annealing process which each iteration loop consists of twenty five iterations and the function value settles down after a long up-down random approach.

(2) Random output data set analysis:

Another approach is that one set of the digitized data is considered as the seed. To compute and identify the merits of each algorithm, a random rotation and translation program was developed and incorporated to simulate the random translation and rotation characteristics. From the utilization of a random number generator [Pre92], the seed points are stochastically translated and rotated to all the possible combinations. For simulation purposes, the following parameters for the transformation are assumed to manipulate the input data points. They are

(A) -50 mm $<$ Random Translation (three orthogonal axes) $<$ 50 mm

(B) -20 degrees $<$ Random Rotation (three Euler angles) $<$ 20 degrees

(C) Gaussian Noise: Mean $= 0$; Variance $= 1$

The selection criteria is based on the reasonable assumption of the patient location

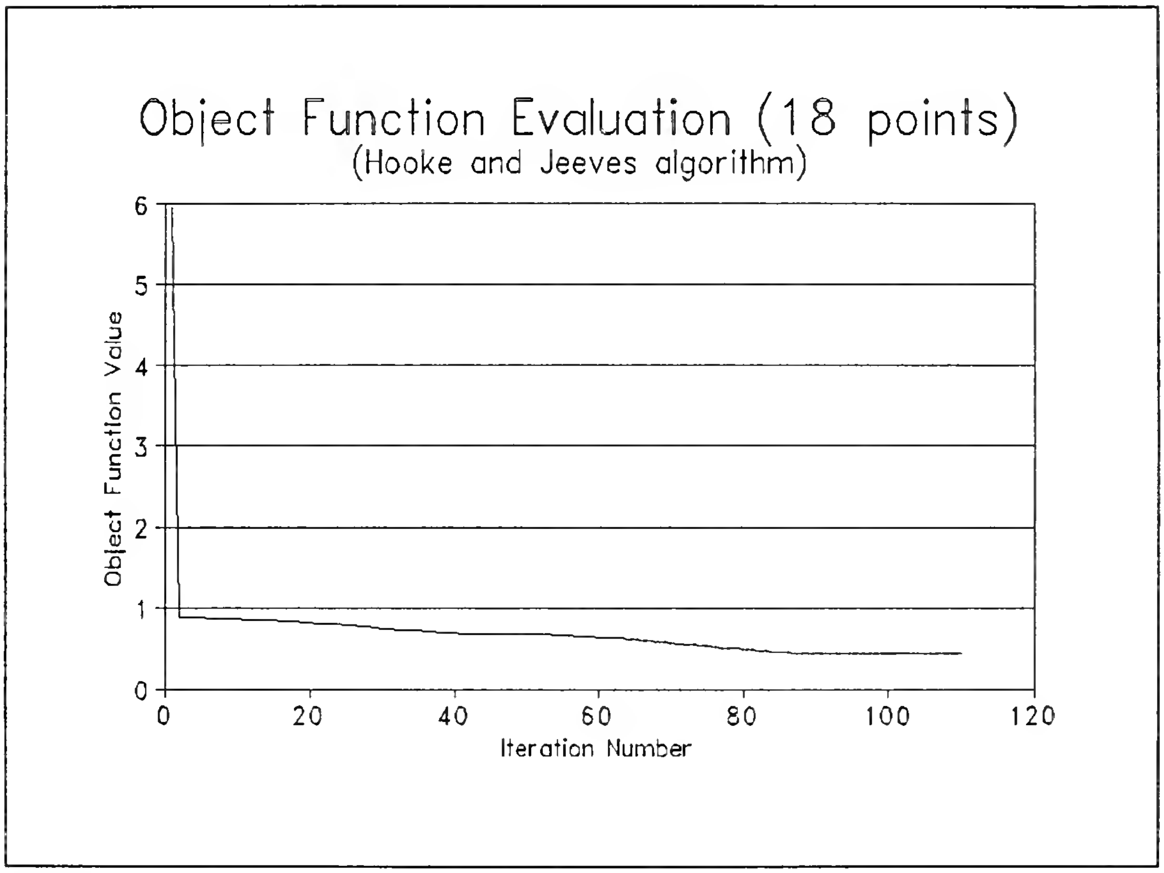


Figure 7.1: Function value changes through each iteration (Hooke and Jeeves algorithm)

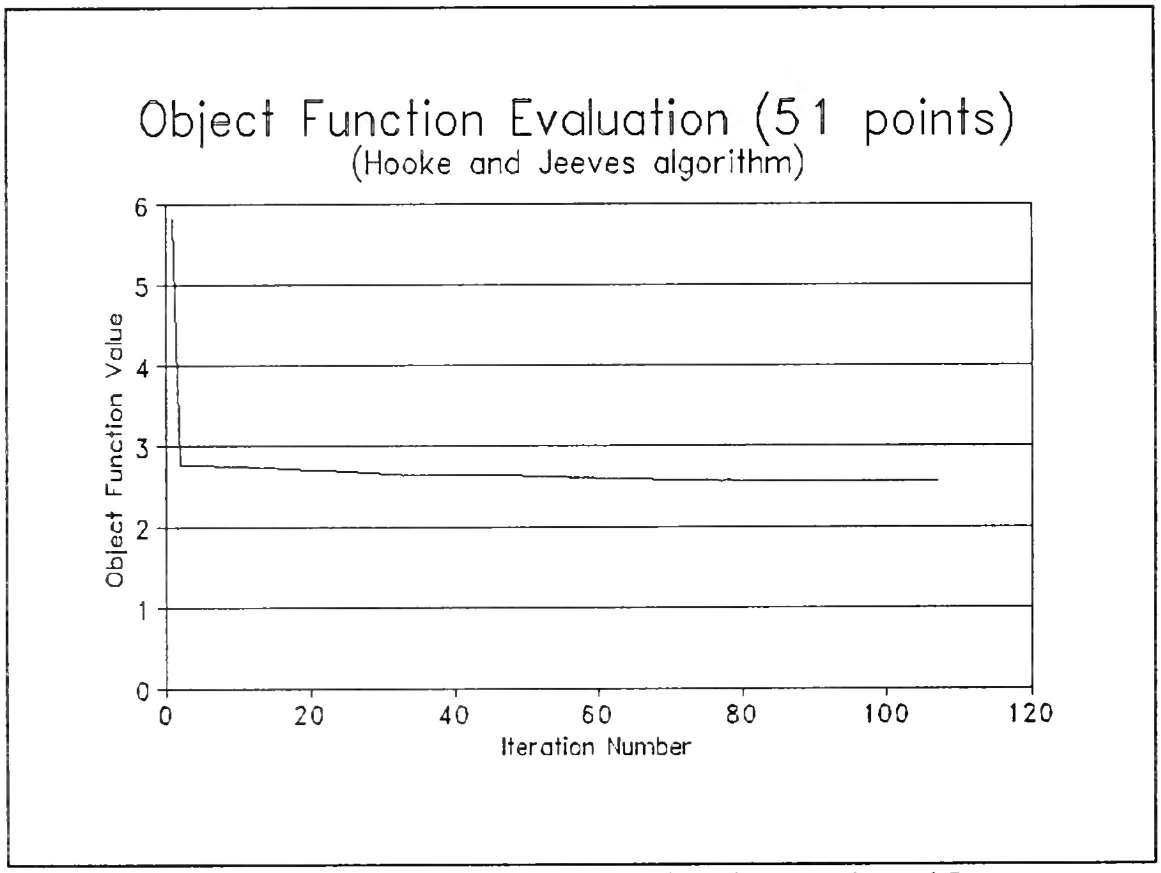


Figure 7-2: Function value changes through each iteration (Hooke and Jeeves algorithm)

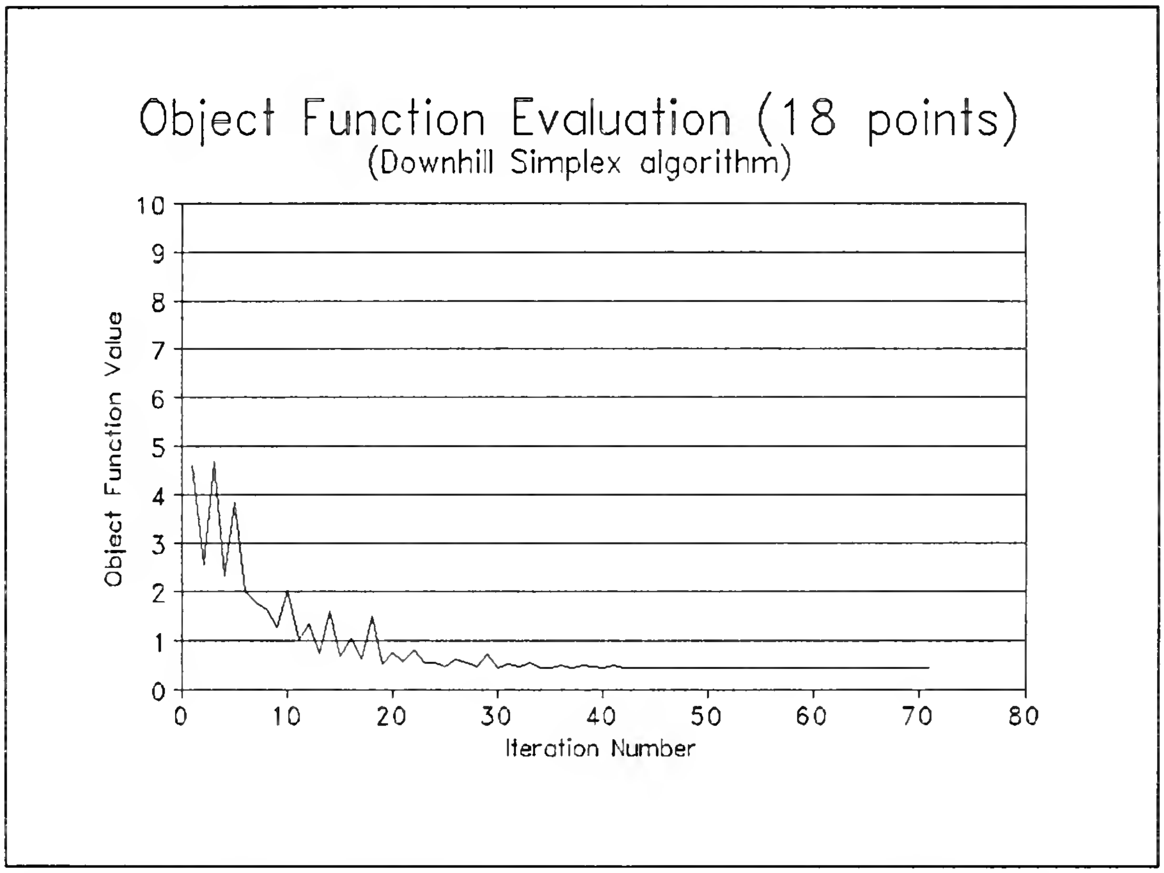


Figure 7-3: Function values changes through each iteration (Downhill Simplex algorithm)

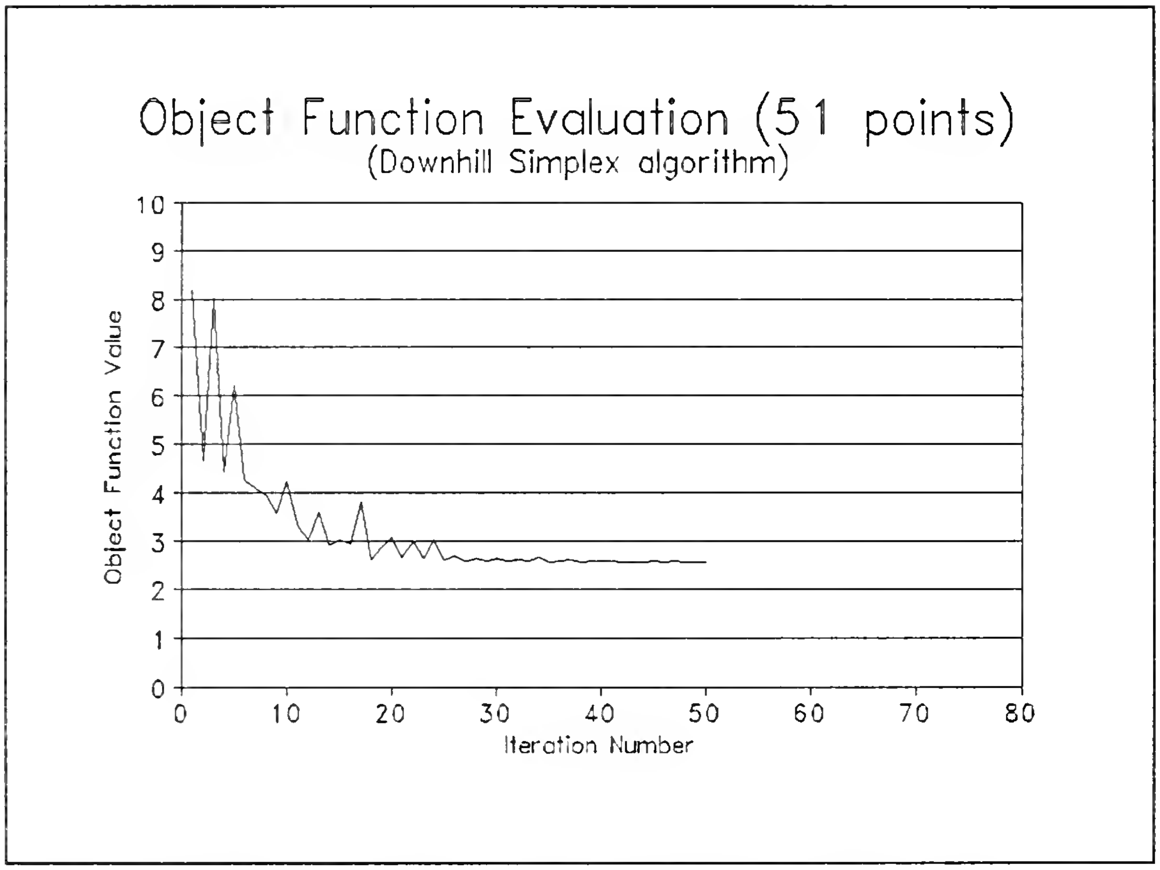


Figure 7-4: Function value changes through each iteration (Downhill Simplex algorithm)

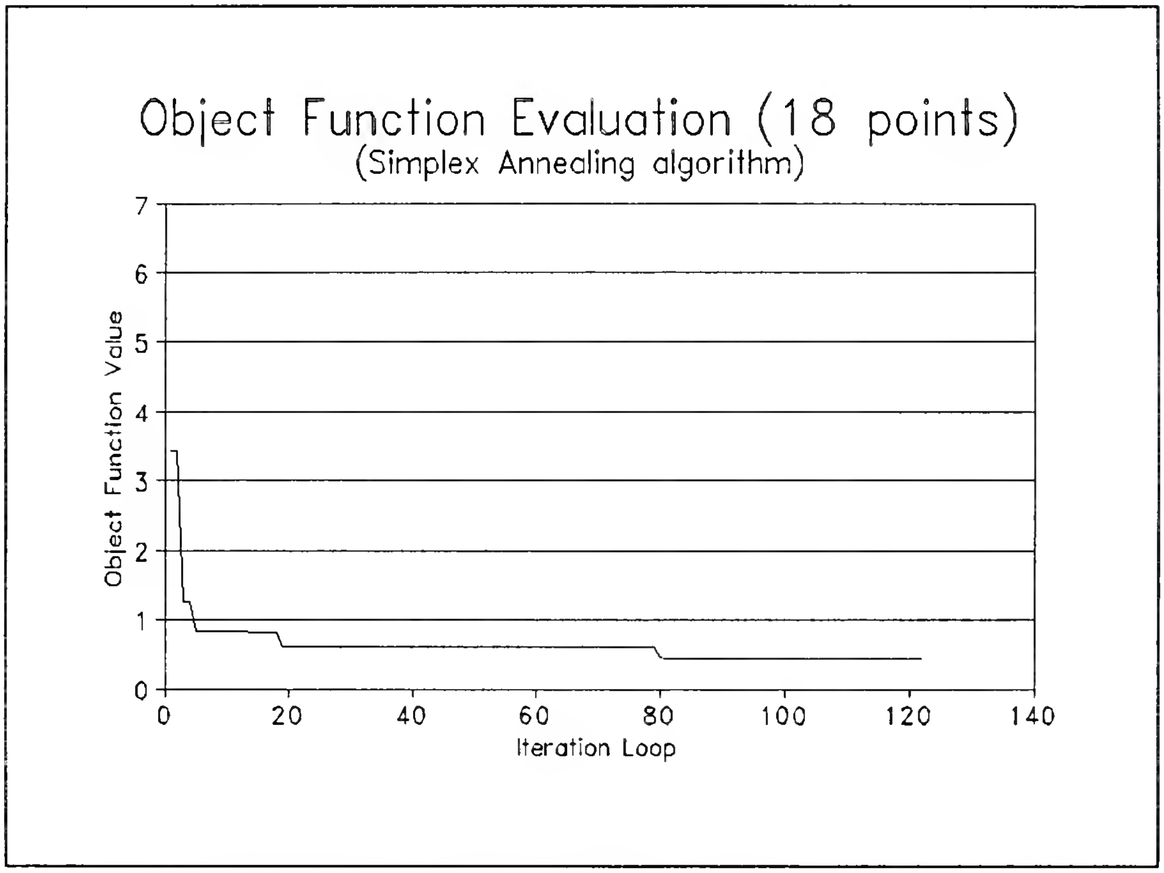


Figure 7-5: Function value changes through each iteration loop (Simplex Annealing algorithm)

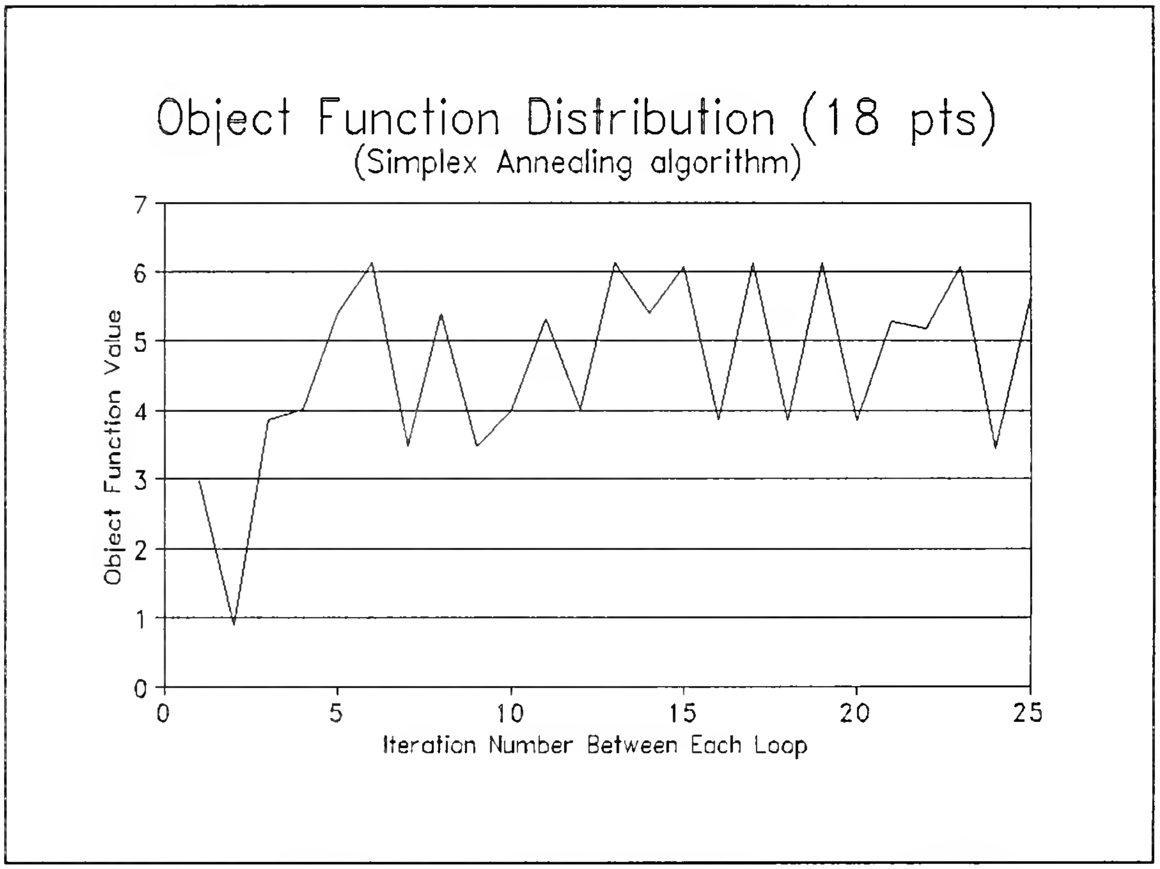


Figure 7-6: The function value changes within each iteration loop of Simplex Annealing algorithm

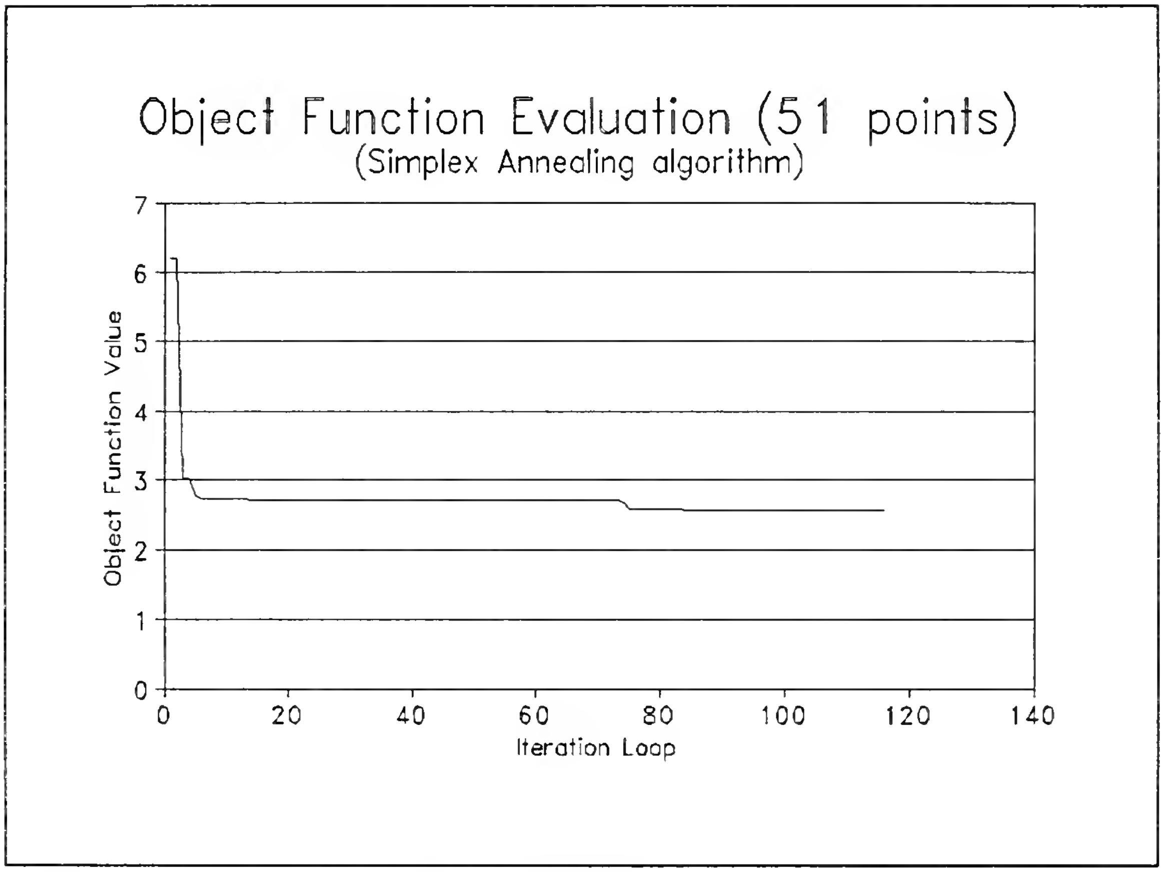


Figure 7-7: Function value changes through each iteration loop (Simplex Annealing algorithm)

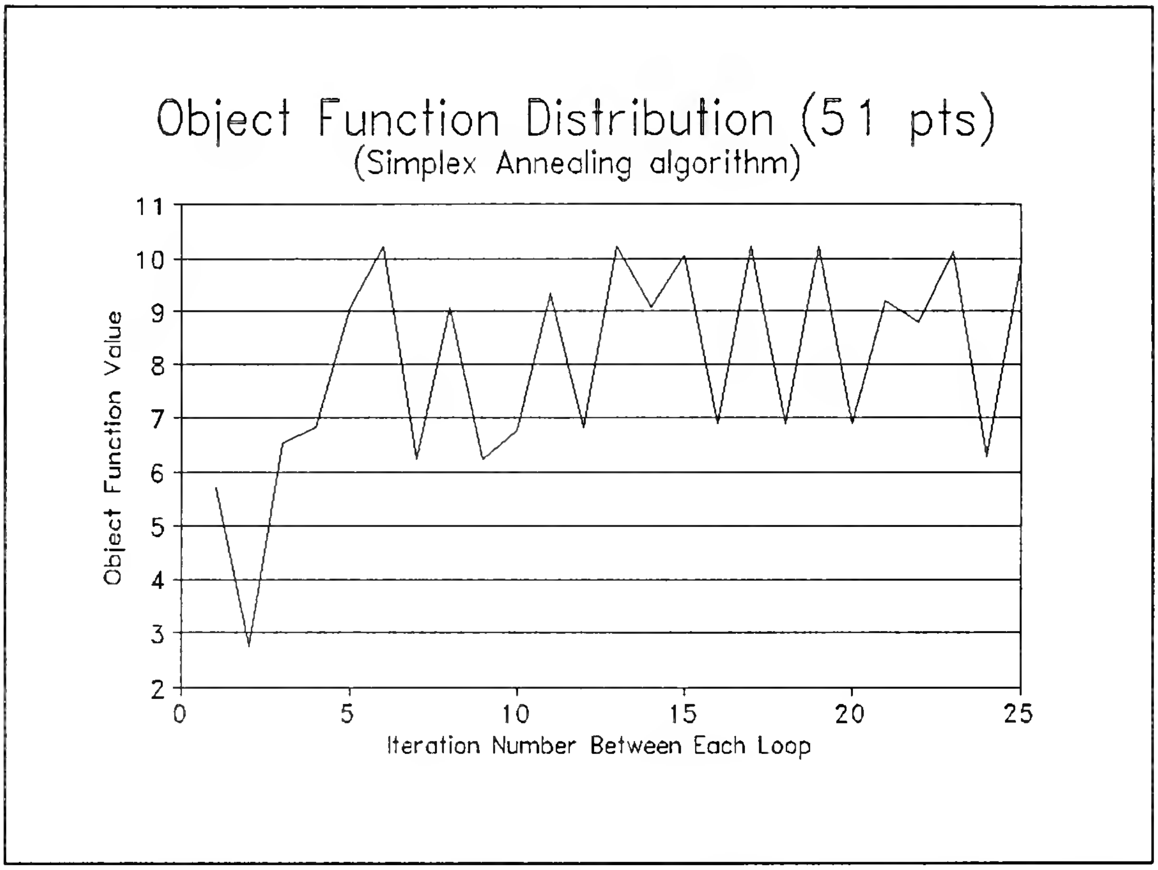


Figure 7-8: The function value changes within each iteration loop (Simplex Annealing algorithm)

Table 7-1: Error Analysis of two sets of 18 3-D data points. A least-square matching is assumed in the object function and the mean error, mean square error and root mean square error in millimeters are calculated from all the data points.

Algorithms	Mean Error (mm)	Mean Square Error (mm^2)	Root Mean Square Error (mm)
SVD	7.620	74.990	8.660
Hooke & Jeeves	4.223	26.608	5.158
Downhill Simplex	4.226	26.707	5.168
Simulated Annealing	4.223	26.624	5.160

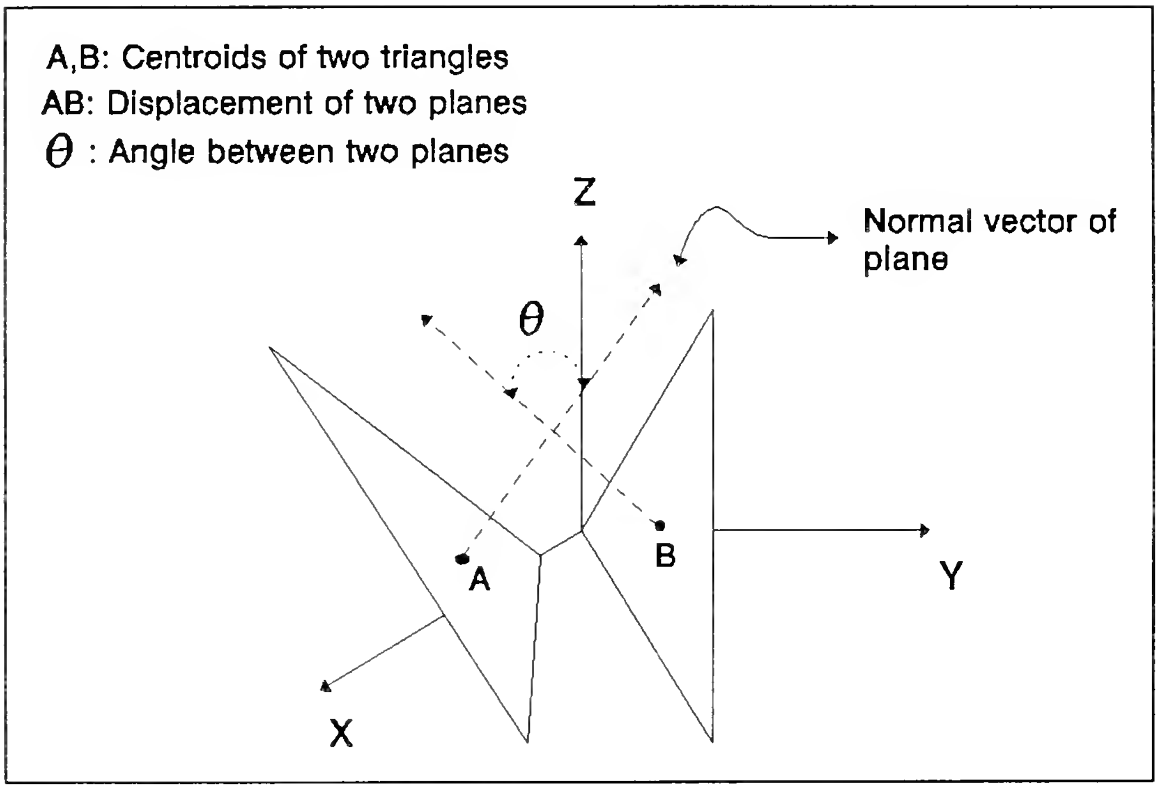
Table 7-2: Error Analysis of two sets of 51 3-D data points. A least-square matching is assumed in the object function and the mean error, mean square error and root mean square error in millimeters are calculated from all the data points.

Algorithms	Mean Error (mm)	Mean Square Error (mm^2)	Root Mean Square Error (mm)
SVD	18.290	380.980	19.520
Hooke & Jeeves	13.937	249.370	15.791
Downhill Simplex	14.041	252.019	15.875
Simulated Annealing	13.941	249.343	15.791

variation in a typical clinical setting. Therefore, one hundred randomly translated and rotated data sets with Gaussian distribution noise are generated. They are all within the margins defined above. The one hundred randomized data sets provide a quantitative basis for the comparison. In order to obtain a better feeling about the target coordinate changes, the following approach is assumed. Instead of analyzing the mean error, mean square error and root mean square error, three non-coplanar coordinates of fiducial markers are incorporated into the data set for verification purposes. The error terms of the target position can be divided as the displacement mean error of two spatial triangles and the angle difference of these two spatial triangle planes (Fig. 7-9). Fig. 7-10 to Fig. 7-17 show the error distribution of the four different algorithms. Notice that three optimization algorithms deliver very close results. A summary of the results of with all four algorithms is shown in Table 7-3.

Phantom Image Tests

Visual based study sometimes delivers better appreciation about how the patient can be realigned. A phantom image test was therefore performed to evaluate the 6-D fitting parameters for the matching results. The patient study started with CT images of two different patient positions and orientations with the phantom head fixed in the BRW localizer (Fig. 7-18). The images were acquired and processed under the image processing program developed at the University of Florida. The correspondent BRW coordinates were retrieved and shown on computer image display canvases. A generalized program was written for the X-11™ system which can display and read in CT images with their BRW coordinate information. A standard right hand Euler frame of


A,B: Centroids of two triangles
AB: Displacement of two planes
θ : Angle between two planes
Z
θ
Normal vector of plane
A
B
Y
X
Figure 7-9: Error definition of target position

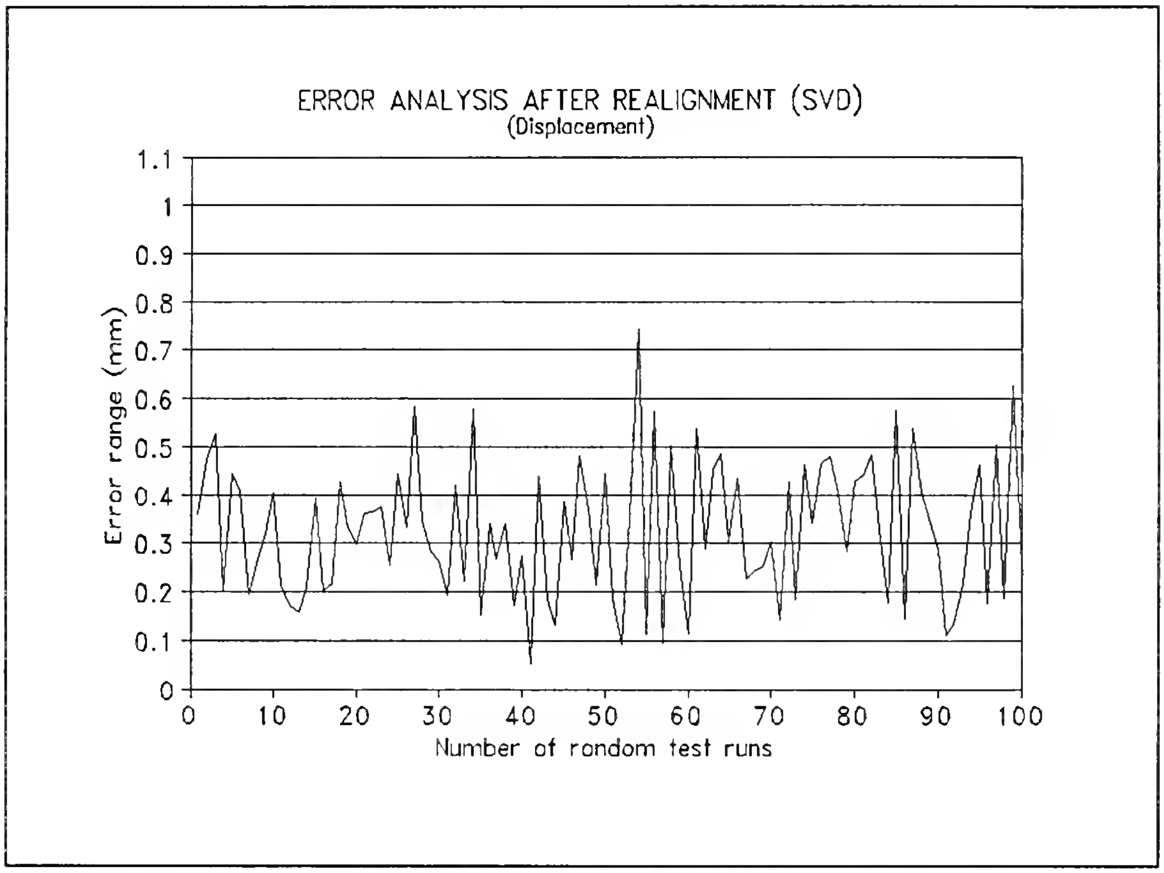


Figure 7-10: Displacement error of target planes (SVD algorithm)

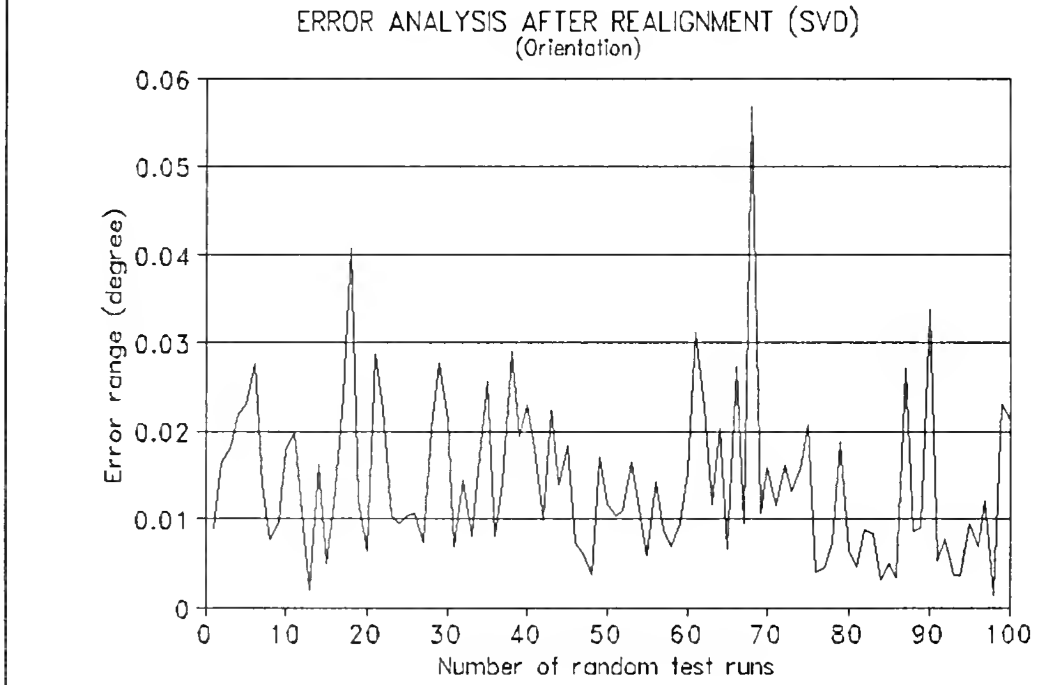


Figure 7-11: Orientation error of target planes (SVD algorithm)

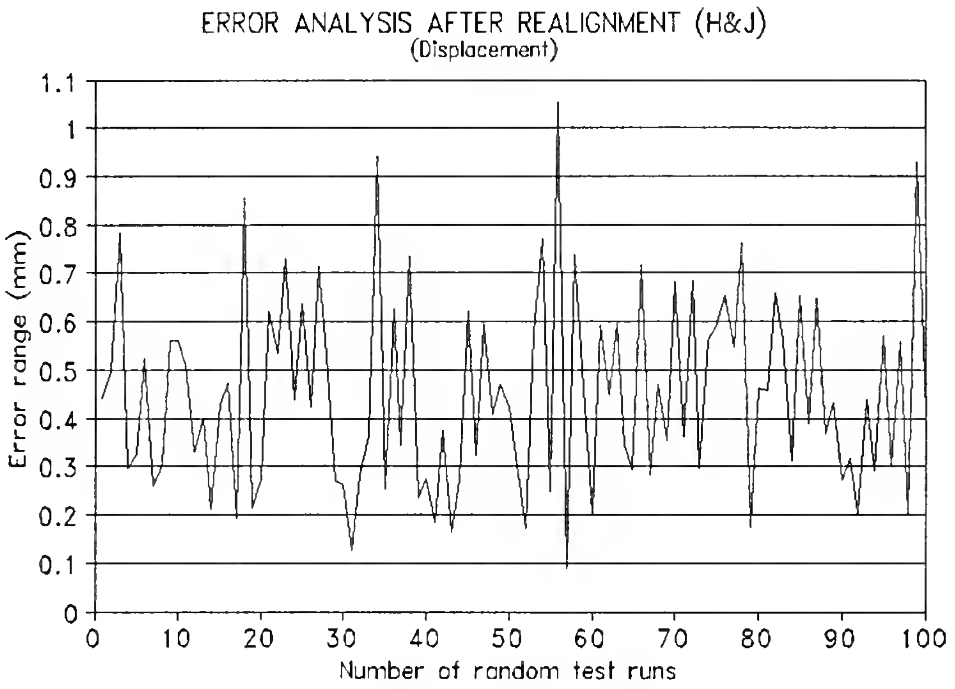


Figure 7-12: Displacement error of target planes (Hooke and Jeeves algorithm)

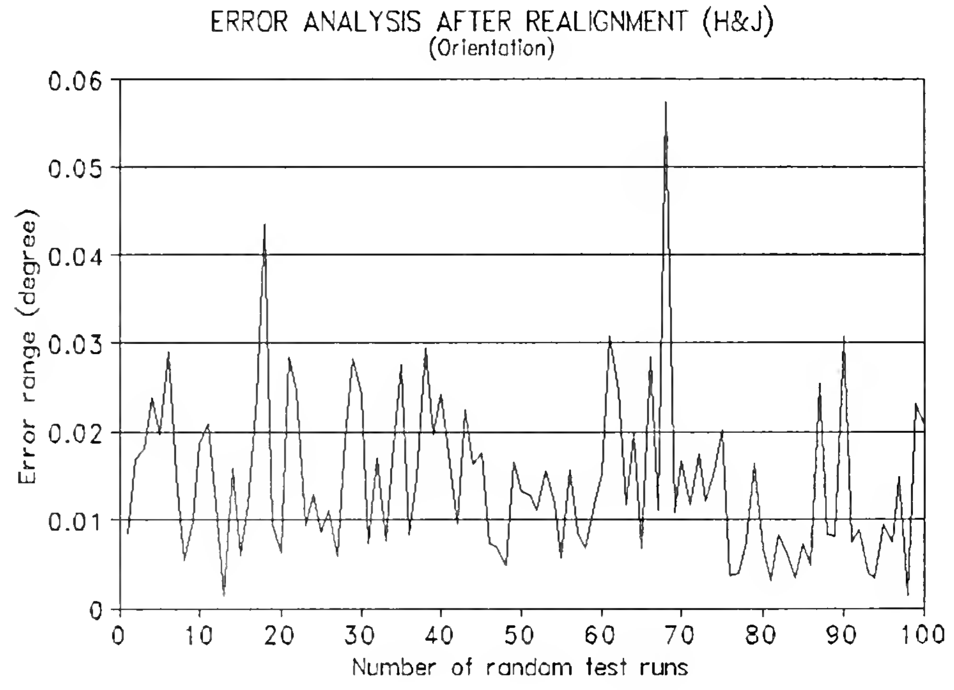


Figure 7-13: Orientation error of target planes (Hooke and Jeeves algorithm)

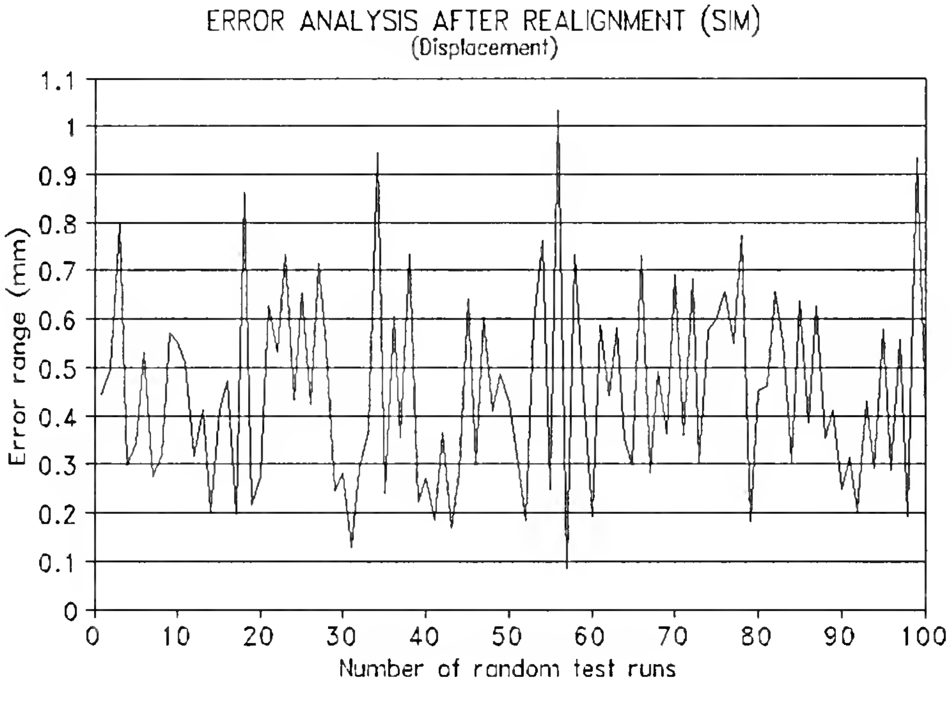


Figure 7-14: Displacement error of target planes (Downhill Simplex algorithm)

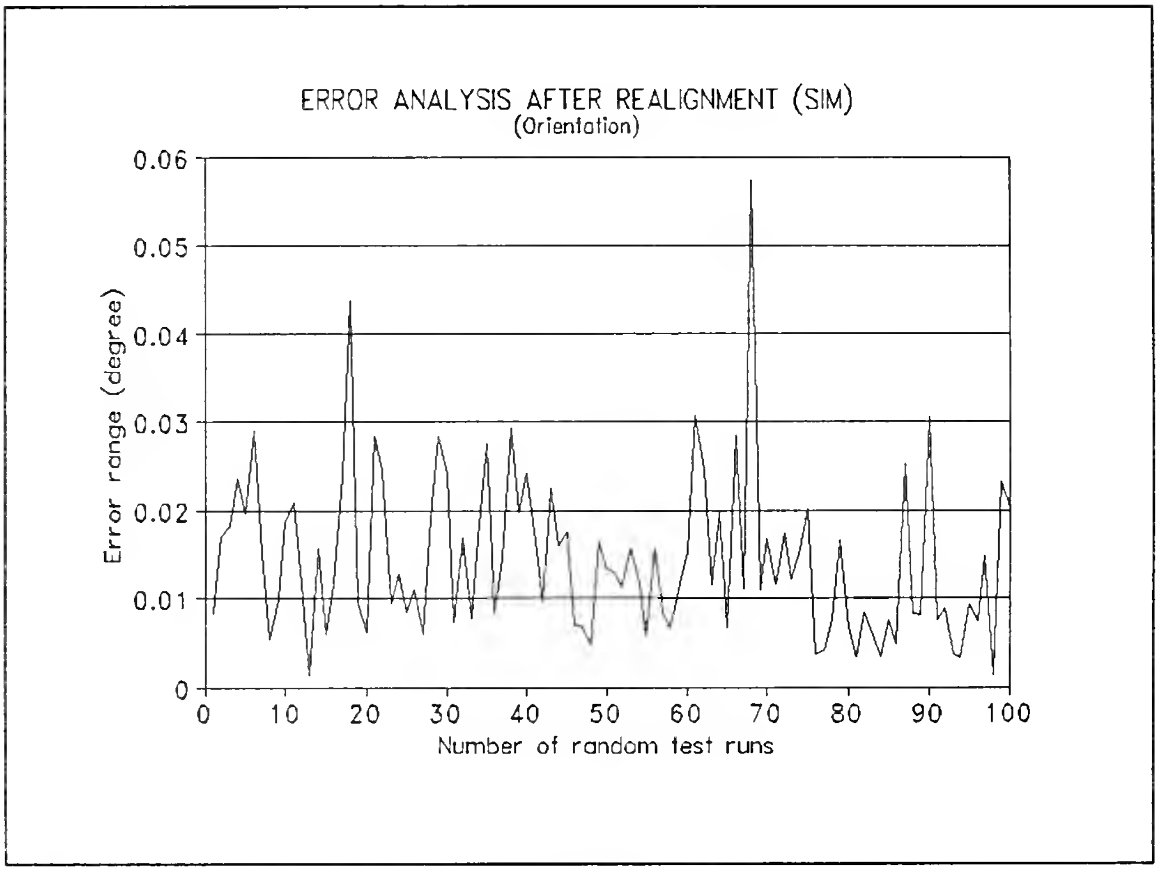


Figure 7-15: Orientation error of target planes (Downhill Simplex algorithm)

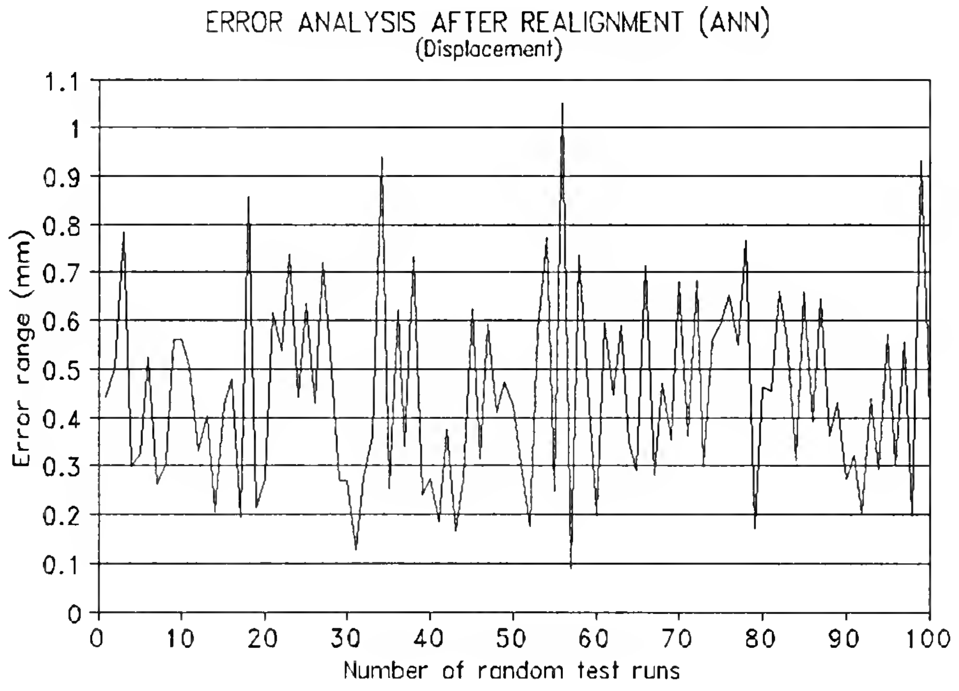


Figure 7-16: Displacement error of target planes (Simulated Annealing algorithm)

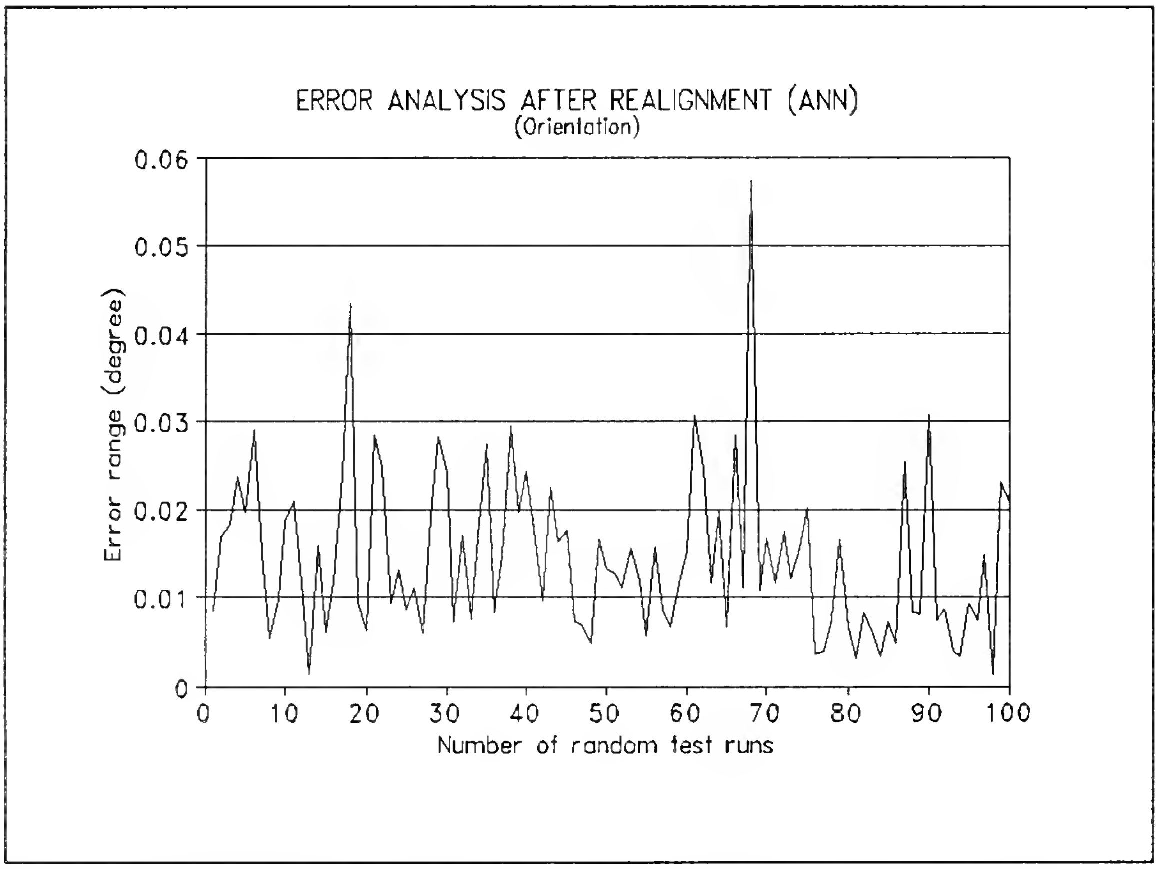


Figure 7-17: Orientation error of target planes (Simulated Annealing algorithm)

Table 7-3: Summary of displacement error and orientation error which are analyzed from one hundred sets of randomly generated data.

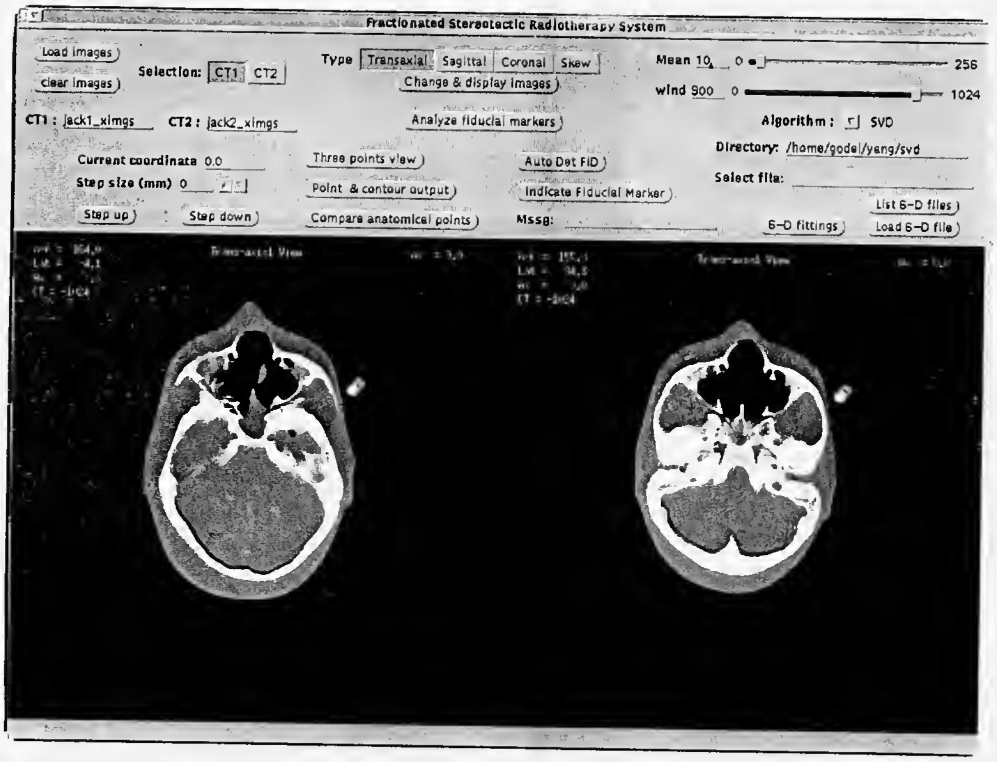
Algorithms	Displacement Mean Error	Orientation Error
SVD	0.4mm±0.15mm	0.01°±0.01°
Hooke & Jeeves	0.4mm±0.19mm	0.01°±0.01°
Downhill Simplex	0.4mm±0.19mm	0.01°±0.01°
Simulated Annealing	0.4mm±0.19mm	0.01°±0.01°

reference is assumed in this study. The positive X-axis is defined towards viewer's eyes. The positive Y-axis is defined towards to right hand side of the screen. The Z-axis is therefore pointed upwards. The effect of rotation on each axis can be seen in Fig. 7-19 to Fig. 7-21. This procedure is to define the rotational axes on the image base and all the transformation parameters are referred to the defined right-hand Euler coordinates.

The image correlation and fitting process begins with the following procedures:

(1) Select and process the three points view in the second set of scanned CT images

By using different CT axial image slices, the patient structure textures can be defined as the reference system for mapping purposes. The midline of the patient facial structure is selected from the three point coordinates to form a proper sagittal cut plane (Fig. 7-22 through Fig. 7-24). This midline usually carries and represents the invariability of the patient surface information. The midline sagittal cut plane should be identical for different patient positions and locations if the three structural points are properly selected and the plane is also adequately reconstructed (Fig. 7-25). Setting the patient midline should re-configure the adequate sagittal plane for patient contour registration. Therefore, patient midline is a solid data base for patient registration, no matter how the patient is positioned or oriented. This selection is highly interactive process which depends on the physician to define the sagittal cut plane. If the plane is perfectly defined, on the image display section of the computer terminal, two identical sagittal planes should be observed. However, this process can never be perfect, so two comparable but not identical sagittal cut planes will be observed after the selection process.



Selection: CT1 CT2

Type: Transaxial Sagittal Coronal Skew

Mean 10 0 256

wind 900 0 1024

CT1 : jack1_ximgs CT2 : jack2_ximgs

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file:

Current coordinate 0.0

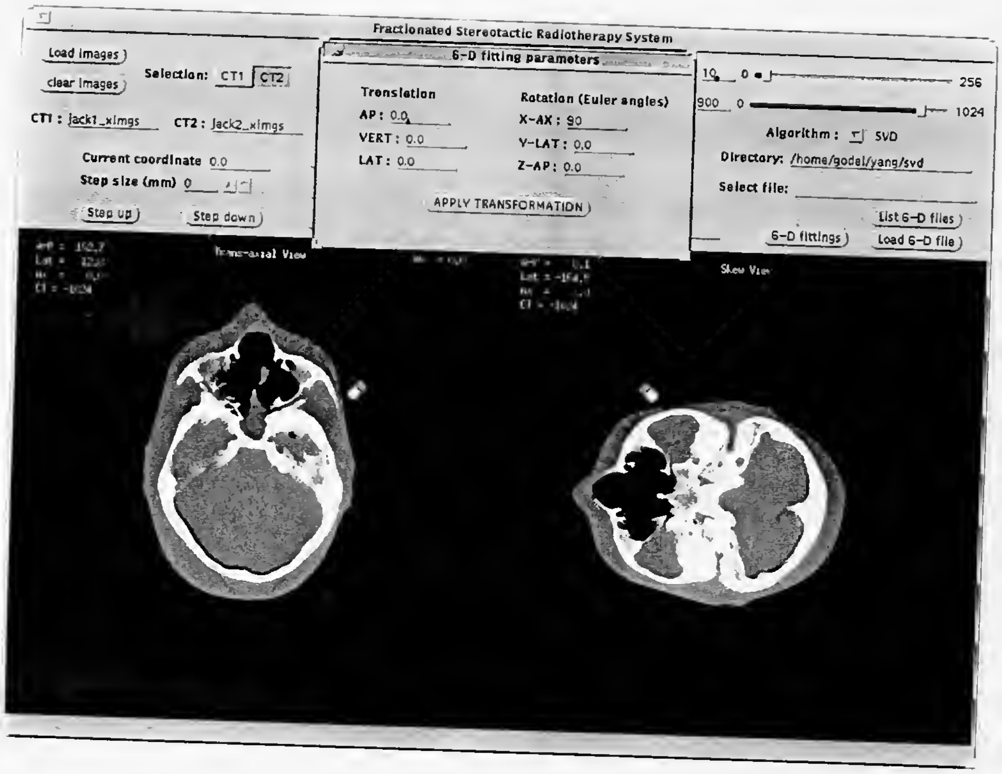
Step size (mm) 0

Mssg:

CT = -1024

CT = -1024

Figure 7-18: Two sets of scanned images in the BRW coordinate system



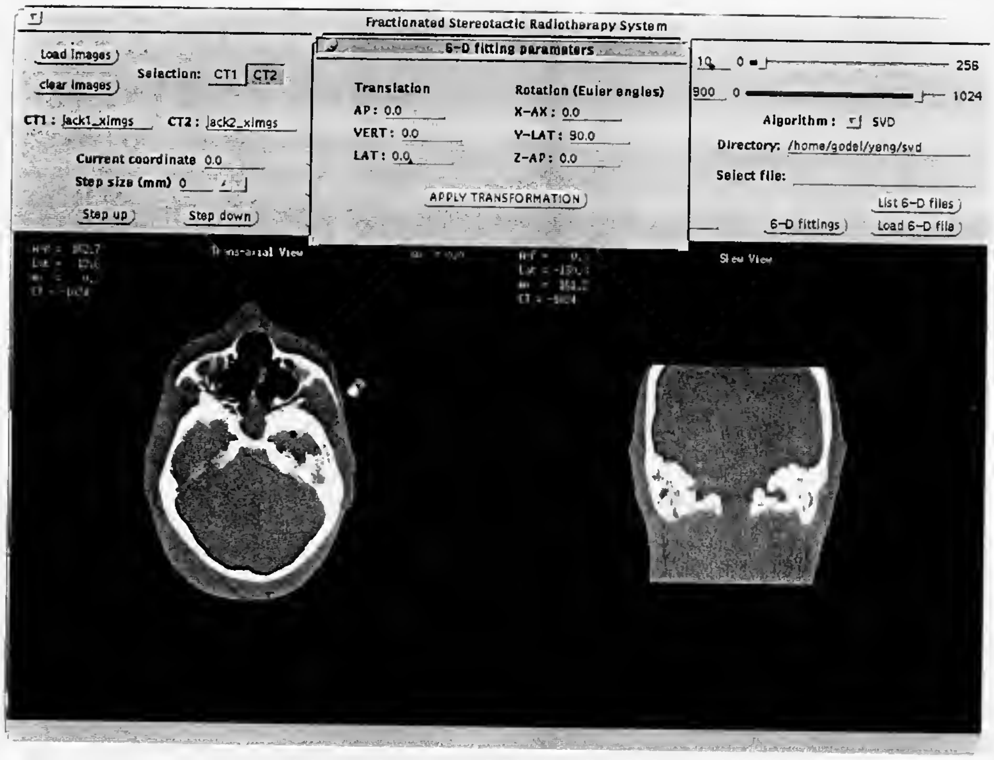
Selection: CT1 [✓ CT2]
CT1 : Jack1_ximgs CT2 : Jack2_ximgs
Current coordinate 0.0
Step size (mm) 0

Translation
AP : 0.0
VERT : 0.0
LAT : 0.0

Rotation (Euler angles)
X-AX : 90
Y-LAT : 0.0
Z-AP : 0.0

10 0 256
900 0 1024
Algorithm : SVD
Directory: /home/godel/yang/svd
Select file:

Figure 7-19: Euler rotation on X-axis



Selection: CT1 CT2

CT1 : Jack1_ximgs   CT2 : Jack2_ximgs

Current coordinate 0.0

Step size (mm) 0

Translation

AP : 0.0

VERT : 0.0

LAT : 0.0

Rotation (Euler angles)

X-AX : 0.0

Y-LAT : 90.0

Z-AP : 0.0

10 0 256

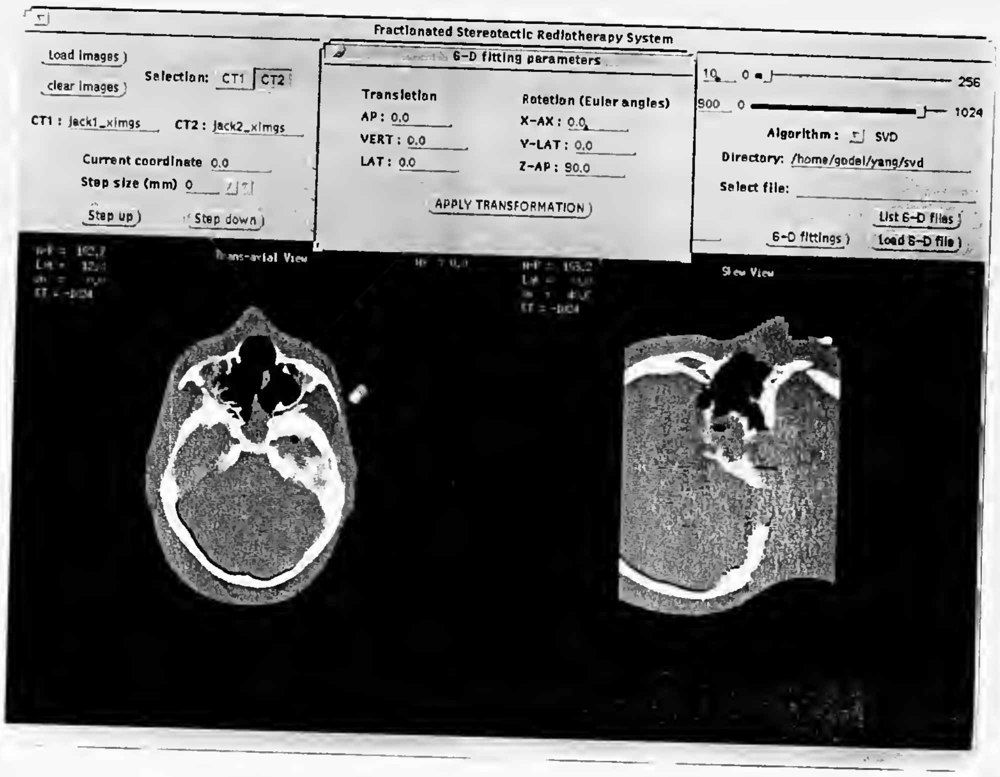
900 0 1024

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file:

Figure 7-20: Euler rotation on Y-axis


Fractionated Stereotactic Rediotherapy System

Load images

clear images

Selection: CT1 [✓ CT2]

CT1 : jack1_ximgs CT2 : jack2_ximgs

Current coordinate 0.0

Step size (mm) 0

Step up

Step down

6-D fitting parameters

Translation

AP : 0.0

VERT : 0.0

LAT : 0.0

Rotation (Euler angles)

X-AX : 0.0

Y-LAT : 0.0

Z-AP : 90.0

APPLY TRANSFORMATION

10 0 256

900 0 1024

Algorithm : SVD

Directory: /home/godel/yang/svd

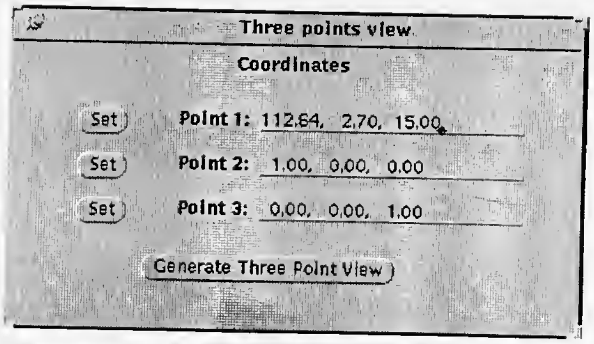
Select file:

List 6-D files

6-D fittings

Load 6-D file

Figure 7-21: Euler rotation on Z-axis


Three points view

Coordinates

Set Point 1: 112.64, 2.70, 15.00

Set Point 2: 1.00, 0.00, 0.00

Set Point 3: 0.00, 0.00, 1.00

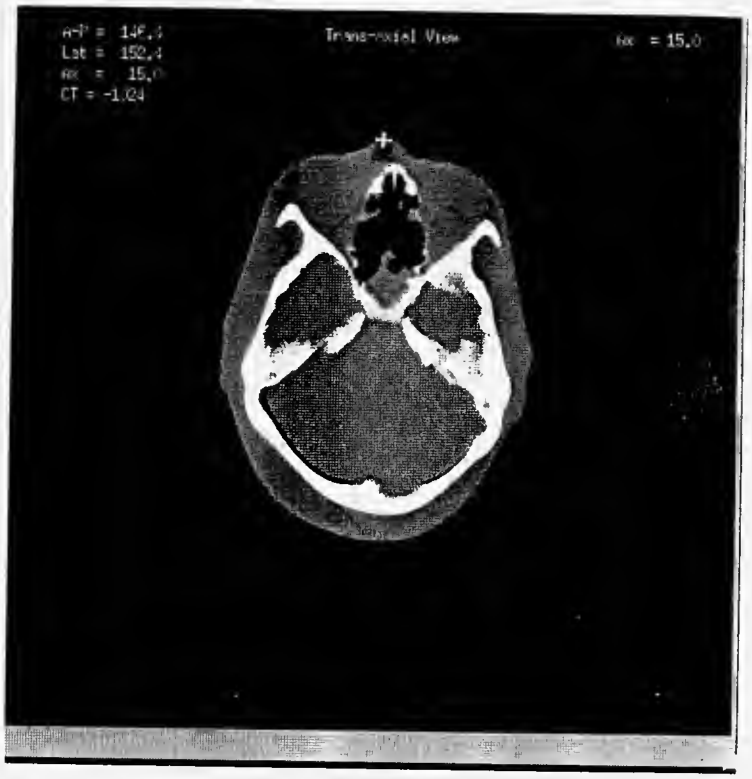
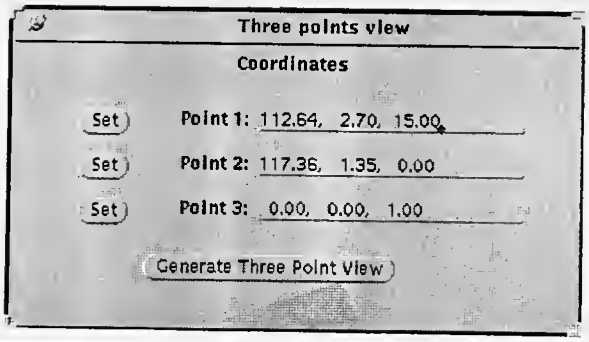


Figure 7-22: Selection of point number one on the midline contour


Three points view

Coordinates

Set Point 1: 112.64, 2.70, 15.00

Set Point 2: 117.36, 1.35, 0.00

Set Point 3: 0.00, 0.00, 1.00

Generate Three Point View

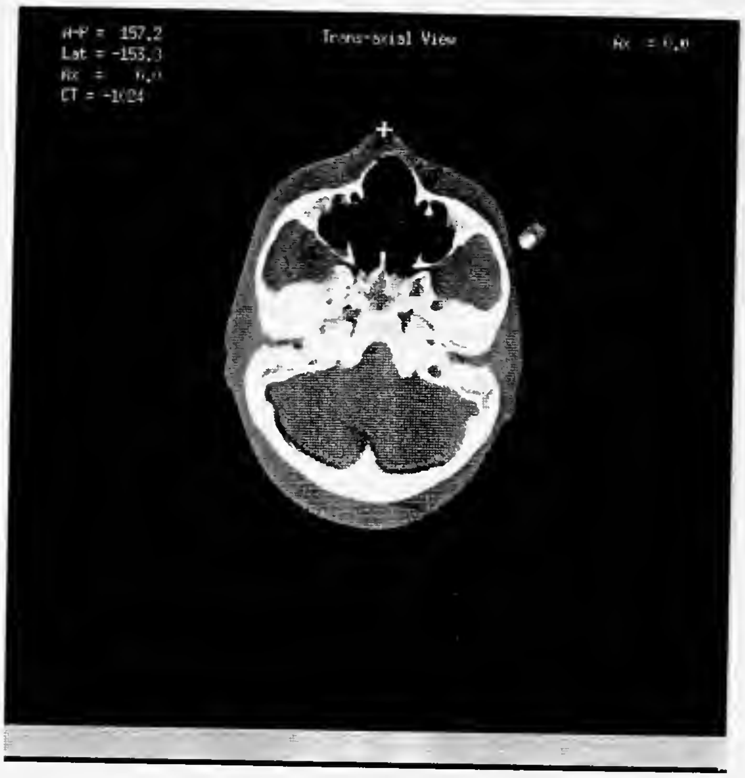
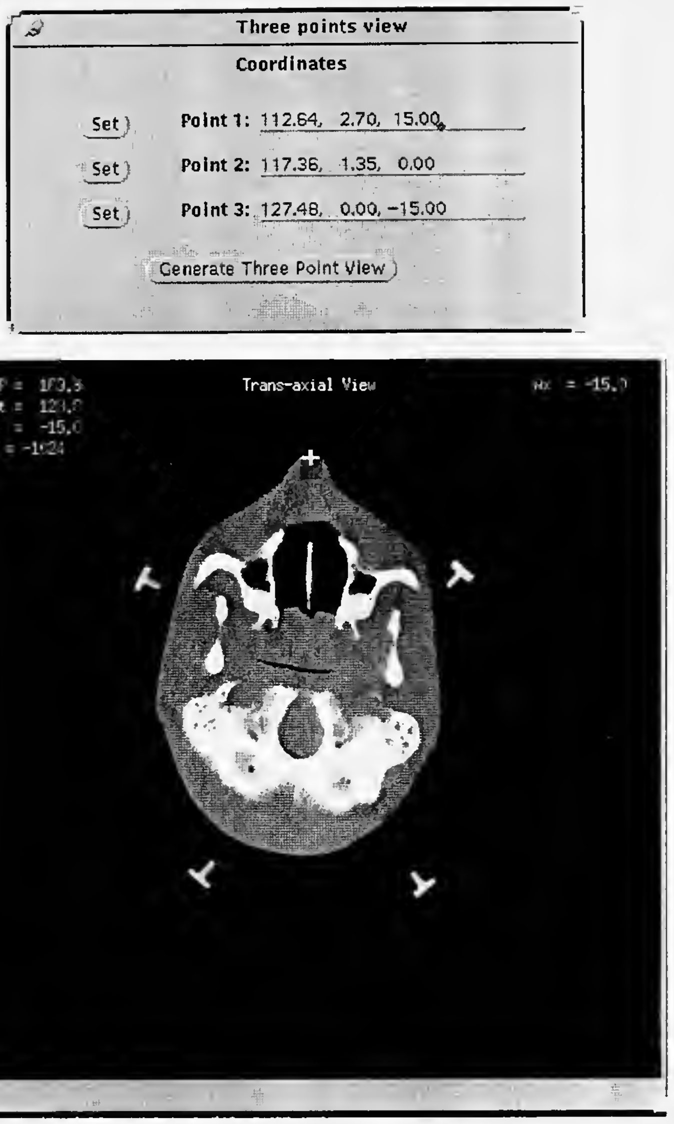


Figure 7-23: Selection of point number two on midline contour


Three points view

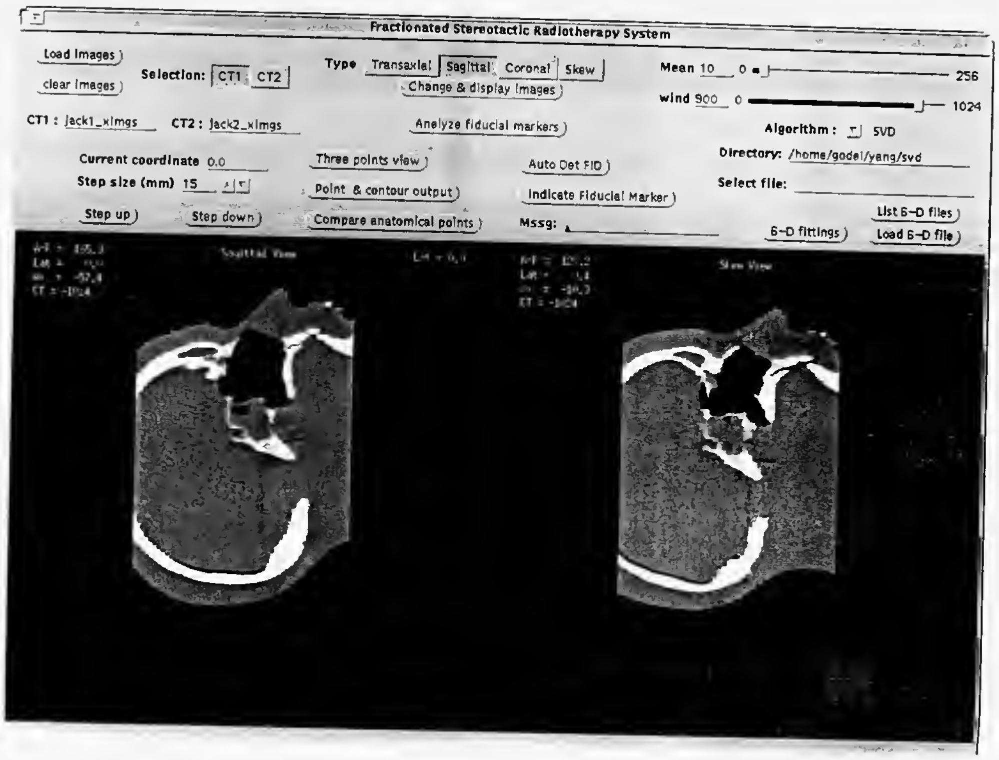
Coordinates

Set Point 1: 112.64, 2.70, 15.00

Set Point 2: 117.36, 1.35, 0.00

Set Point 3: 127.48, 0.00, -15.00

Figure 7-24: Selection of point number three on midline contour



Selection: CT1 CT2

CT1 : jack1_ximgs CT2 : jack2_ximgs

Current coordinate 0.0

Step size (mm) 15

Mean 10 0 256

wind 900 0 1024

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file:

Mssg:

Figure 7-25: Reconstructed sagittal cut plane from three selective points, shown as skew view

(2) Selection of the midline contour and the proper anatomical points

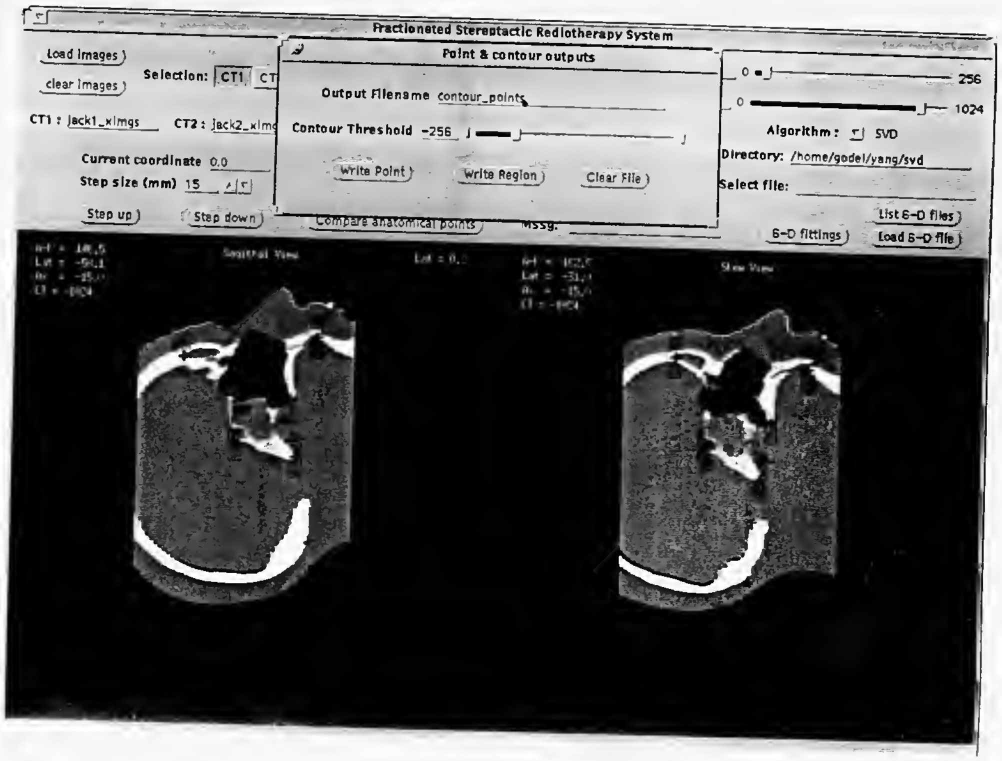
In addition to midline information rigid anatomical points are chosen to incorporate into the data sets. On the patient surface, there are anatomical points such as the tragus point, eye commissure points or other solid distinguishable points which carry suitable information important for the mapping matrix calculation. Those points need to be registered by the physician, as shown in Fig. 7-26. In this case, the phantom head does not have obvious anatomical points except for the bony structures. The tips of the various bony structures are chosen to fulfill the anatomical information requirement.

(3) Form a proper data set for calculation

There are two different structural information files that describe the patient locations, i.e., the midline contour points and the facial anatomical points. They both represent important geometrical data which delineates patient position and orientation. To reformat these information files, the process which was described in Chapter 4 is utilized. First, sort the data files into increasing axial slice coordinates (in the display mode, it is the Euler X coordinate). Second, cubic spline refit the data files to form an equal number of points to minimize finite resolution errors caused by the CT/MR images. Third, with the proper linear weighting function, these two information files are re-organized into a data set which contains the total number of data points and the two sets of 3-D coordinates.

(4) Apply the mathematical mapping function to calculate the transformation matrix

Different algorithms are applied to calculate the required transformation matrix for



Selection: CT1

CT1 : jack1_ximgs CT2 : jack2_xim

Current coordinate 0.0

Step size (mm) 15

Output Filename contour_points

Contour Threshold -256

0 256

0 1024

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file:

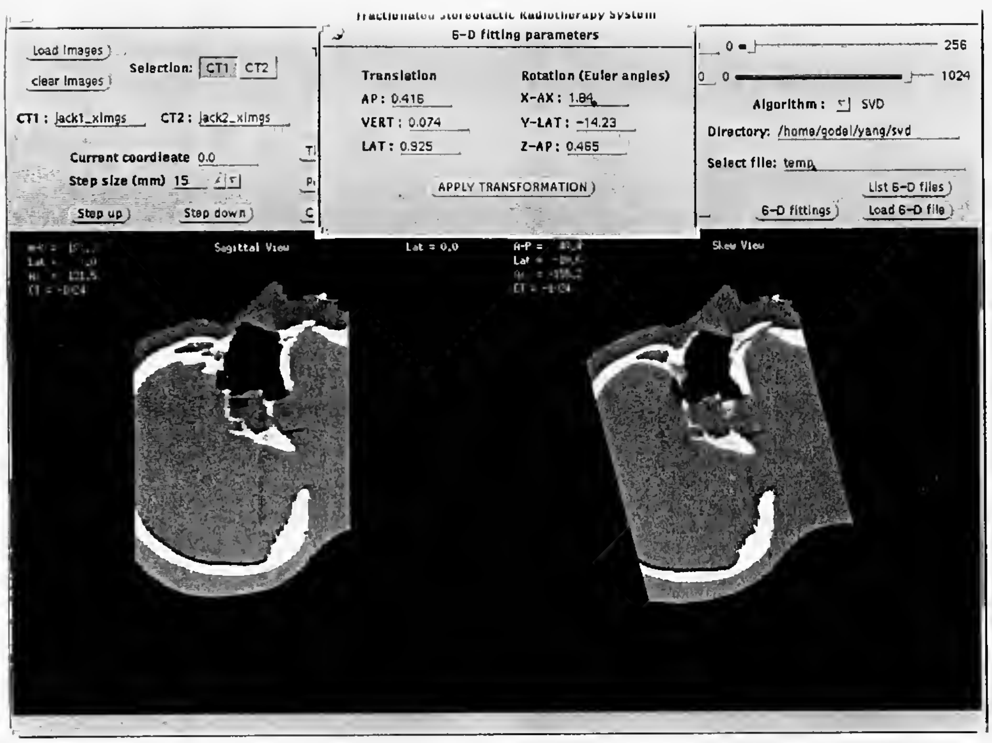
Mssg:

Figure 7-26: Selection of contour line and the anatomical points

patient realignment. The calculation is accomplished in a command mode which is a typical in the C programming environment. The output file is saved and retrieved for the manipulation of the CT/MR images. Since four different algorithms (SVD, Hooke and Jeeves, Downhill Simplex, Simulated Annealing methods) are used to perform the calculation, their results can be compared and analyzed at the same time. The 6-D fitting parameters are implemented using commands generated by window functions and the second set of images can be transformed for further study. The 6-D fitting parameters as well as the transformed second set of image is displayed in Fig. 7-27 and the corresponding bony structures after transformation on the right display canvas are observed.

(5) After proper translation and rotation manipulation, the two image sets are compared to identify the target offset

After the transformation on the second image data set with the calculated matrix values, a new skewed image set will be displayed on the canvas in the final patient treatment location. Now the two different scanned images can be correlated since the final treatment location of the second image set is presented with respect to the initial image set. Structural anatomical points can be selected to compare the offset. Again, the results will be verified by the physician. Especially for the critical structure points, the calculated offset offers a quantitative comparison for the realignment process for patient fractionated treatment location. Fig. 7-28 and Fig. 7-29 show the vector errors calculated from two anatomical bony structures. Notice that the CT pixel size is about 0.67 mm, therefore 2mm error represents 3 pixels difference.



Selection: CT1 CT2

CT1 : Jack1_ximgs CT2 : Jack2_ximgs

Current coordinate 0.0

Step size (mm) 15

Translation

AP : 0.416

VERT : 0.074

LAT : 0.925

Rotation (Euler angles)

X-AX : 1.84

Y-LAT : -14.23

Z-AP : 0.465

0 256

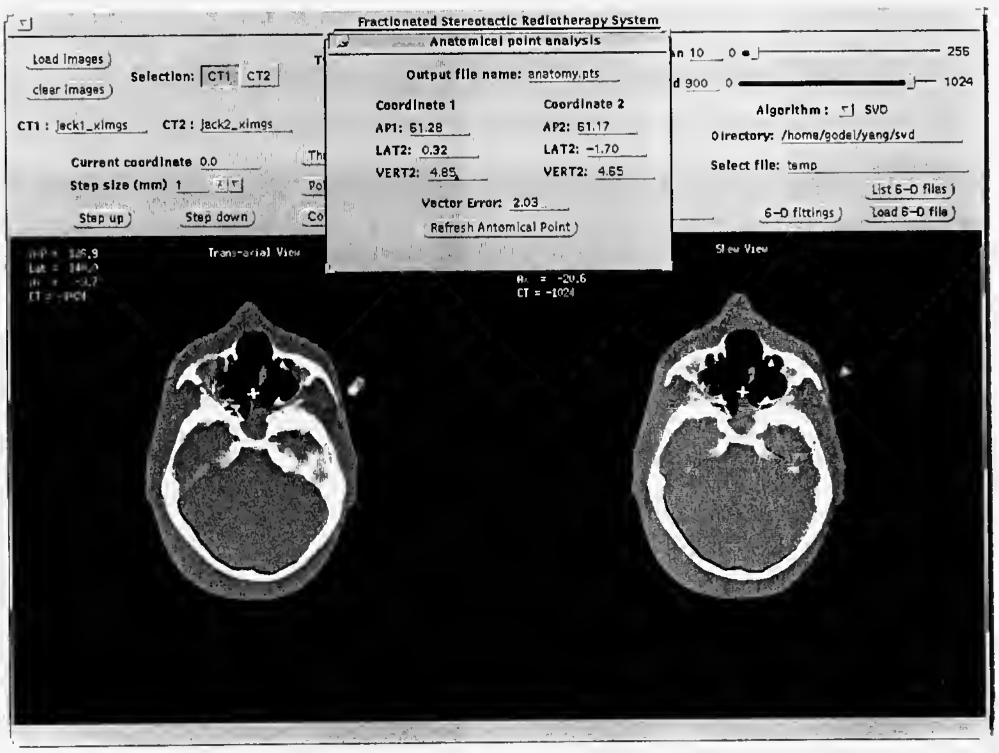
0 1024

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file: temp

Figure 7-27: Display of the original and transformed image sets. The bony structure should be close after applying the transformation matrix



Fractionated Stereotactic Rediotherapy System

Anatomicel point analysis

Output file name: anatomy.pts

Coordinate 1
AP1: 61.28
LAT2: 0.32
VERT2: 4.85

Coordinate 2
AP2: 61.17
LAT2: -1.70
VERT2: 4.65

Vector Error: 2.03

Selection: CT1 CT2

CT1 : jeck1_ximgs CT2 : jack2_ximgs

Current coordinate 0.0

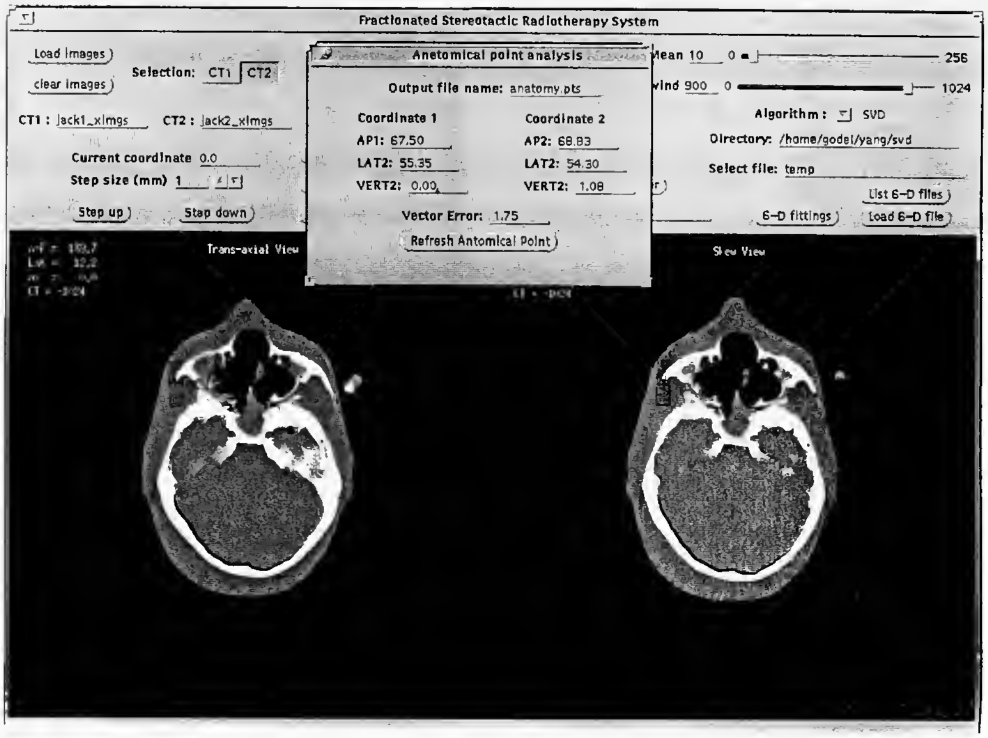
Step size (mm) 1

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file: temp

Figure 7-28: Comparison of anatomical structure on the original image and the transformed image (set 1)


Fractionated Stereotactic Radiotherapy System

Selection: CT1 CT2

CT1 : Jack1_xImgs CT2 : Jack2_xImgs

Current coordinate 0.0

Step size (mm) 1

Anetomical point analysis

Output file name: anatomy.pts

Coordinate 1	Coordinate 2
AP1: 67.50	AP2: 68.83
LAT2: 55.35	LAT2: 54.30
VERT2: 0.00	VERT2: 1.08

Vector Error: 1.75

Mean 10 0 256

Wind 900 0 1024

Algorithm : SVD

Directory: /home/godel/yang/svd

Select file: temp

Figure 7-29: Comparison of anatomical structure on the original image and the transformed image (set 2)

3-D Camera System Data Tests

For the correlation of real digitized 3-D data, the Northern Digital OPTOTRAK™ 3-D camera system was utilized to collect the real time phantom data points. For the translation verification, known values of the translation displacements are implemented to study the vector errors. Five different experiments are performed to estimate the translation precision and the results are shown in Table 7-4. Three infrared LED markers were also placed on the phantom head for verification purposes. These three markers generate a geometrical center and two normal vectors which are described in Fig. 7-9. For testing purposes, only 3 mm and 5 mm translational distances on three orthogonal axes are selected to analyze the acquired LED data. The average results for displacement error and orientational error of all the algorithms are shown in Table 7-5.

For the experiments to calculate rotational parameters, a phantom model shown in Fig. 7-30 is created to verify the calculations. This phantom base provides a two dimensional rotation mechanism with separate rotating axes. With three infrared LED markers on the phantom head, the initial location and the final location are recorded as well as the known rotational angles on one fixed axis.

In the determination of the correlation matrix between multiple anatomical points and the midline contour, the same phantom base is also utilized. A digitizer probe with 25 infrared LED markers is applied to read in 3-D information. The patient contour midline is selected and the anatomical structural points are chosen. The pseudo isocenter is calibrated at the rotating point of the base plate which is attached to the phantom head. Known values of rotation angles are also pre-defined to move the phantom head with a

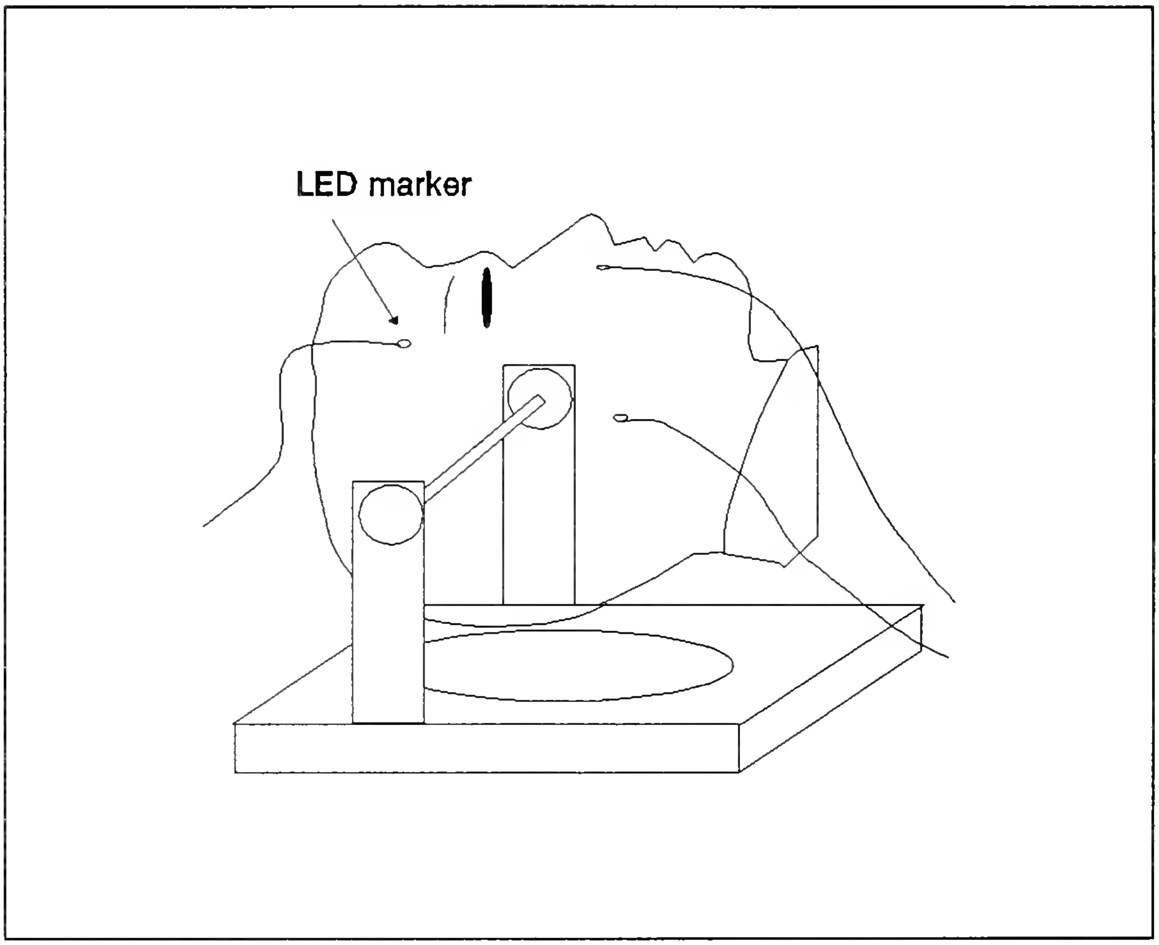


Figure 7-30: The head phantom for 3-D camera system tests

Table 7-4: Error Analysis of two sets of digitized 3-D data points with known translation vectors. A least-square matching is assumed in the object function and the calculated results are displayed.

Test conditions		Calculated results (mm)			
Translation		X	Y	Z	Vector error
exp1	3 mm (X)	2.987	0.091	0.120	0.15
exp2	3 mm (Y)	0.102	3.047	0.083	0.14
exp3	3 mm (Z)	0.121	0.095	2.963	0.16
exp4	3 mm (X,Y,Z)	3.12	2.92	3.15	0.21
exp5	5 mm (X,Y,Z)	5.26	4.90	4.87	0.31

Table 7-5: Error Analysis of infrared LED markers attached to the phantom head. The LED location simulates the bite plate system which will be incorporated into the verification procedure.

Test conditions		Measured and calculated results of LED location (mm, degrees of normal vectors)	
Translation		Average displacement error of LEDs	Orientation error of normal vectors formed by LEDs
exp1	3 mm (X,Y,Z)	(0.097, 0.101, 0.088) (X,Y,Z)	0.024°
exp2	5 mm (X,Y,Z)	(0.112, 0.107, 0.104) (X,Y,Z)	0.045°

known value. Then the initial data set and rotated data set are both retrieved and saved into two files; the correlation procedure starts with four different calculation algorithms. The following Table 7-6 shows the best fit of the two data point sets, therefore, the error analysis can be performed.

The above table only displays experimental and calculated results on the Z rotational axis since it is the primary rotational axis for the experimental purpose. The error analysis on the other rotational axes are also executed. The maximum error around the Z axis 90 degree rotation is +0.596 degree on the X axis, which is the inherent error introduced by the input device (camera digitization process). The difference of angle is from the digitizing procedures which human errors contribute occur in the data collecting procedures.

Three separated infrared LED markers were placed on the phantom head in order to represent and simulate the bite plate verification system which will be used in the future. The same experiment of known rotational value is performed again to evaluate the LED locations in 3-D space. After translation and rotation manipulation, the three LEDs should move back to the original location to illustrate that the repositioning calculation delivers the correct 6-D fitting parameters. Phantom studies show that the experimental errors are all within tolerance. Therefore, the evaluation of treatment location can be fulfilled anytime during the fractionated high precision treatment procedures and the repositioning errors can be also analyzed whenever the motion of patient is suspected.

Table 7-6: Error Analysis of two sets of digitized 3-D data points. A least-square matching is assumed in the object function and the calculated results are displayed.

	Experimental Conditions (rotation on Z axis)				
Algorithms	-5.3°	-10.0°	-20.0°	-90.0°	Maximum error
SVD	-5.375	-9.867	-20.036	-90.596	0.596°
Hooke & Jeeves	-5.305	-10.026	-20.008	-90.567	0.567°
Downhill Simplex	-5.311	-10.021	-20.002	-90.557	0.557°
Simulated Annealing	-5.310	-10.020	-20.005	-90.567	0.567°

Table 7-7: Error Analysis of infrared LED markers attached to the phantom head. The LED location simulates the bite plate system which will be incorporated into the verification procedure.

Test conditions		Measured and calculated results of LED location (mm, degrees of normal vectors)	
Rotation (Z axis)		Averaged Displacement error of LEDs	Orientation error of normal vectors formed by LEDs
exp1	5°	0.039 (1*) 0.085 (2) 0.114 (3)	0.126°
exp2	10°	0.112 (1) 0.108 (2) 0.129 (3)	0.134°
exp3	20°	0.060 (1) 0.173 (2) 0.116 (3)	0.176°

* : Represent the number of the LED markers

Correlation of CT image and 3-D camera data sets

The last step in testing the accuracy of the integrated system is to correlate the CT data set to the real time digitized data set in a phantom testing procedure. First, the phantom image is scanned with all the necessary fiducial markers inside and outside the phantom head. The surface fiducial markers and the target points are then obtained in the CT coordinates which are obtained from the time of setup. These images are processed using modified radiosurgery software developed at the University of Florida to form three reconstructed orthogonal image sets without using the radiosurgery CT localizer. The target isocenter location and the other treatment parameters can be calculated from the standard UF radiosurgery treatment procedure. After the treatment planning process is finished and the isocenter, collimator selection, treatment table angle and gantry rotation angle are determined, the isocenter coordinate of the target volume in the scanned CT coordinate system is translated to become the absolute origin of the treatment coordinate system. All the 3-D coordinates of the surface and anatomical points should be referred to this updated absolute origin after this isocenter shift manipulation. The isocenter of the target volume is now the absolute origin in the treatment coordinate system (i.e., LINAC coordinate system). Since the 3-D real time camera system is also calibrated to the machine isocenter position, all the digitized points contain the 3-D coordinates with respect to the isocenter location.

Secondly, the same phantom surface contour line and the critical anatomical points are selected to form 3-D data sets which describe the specific information of the phantom at the time of treatment setup. The 3-D digitized points are organized to form a complete data set for correlation process. This correlation is determined by mathematical methods and the 6-D fitting parameters are calculated when both the 3-D data sets are ready. Those parameters are used to reposition the patient treatment location and the target location is verified from the digitized probe again. The treatment is then engaged. The subsequent fractionated schedule proceeds with the same methodology and the patient location can be verified at any time. The process can be described as follows:

(1) The solid phantom with contrast is scanned with CT image modality and the image data set is saved as the standard reference information for correlation purposes.

(2) The phantom surface contour line and the specified anatomical points are chosen to represent the rigid model of the phantom head and a 3-D data set with proper numerical manipulation is formed. A three point sagittal cut plane of the phantom base is retrieved as a standard reference for selection of the contour line. The image data set is sorted, spline fitted and weighted which then constructs the basic 3-D information. This curved contour line and other anatomical points become the information source which characterizes the tumor location (isocenter).

(3) The same phantom geometry will be positioned under the 3-D camera system with known translation and rotational angles. The same contour line is retaken with the real time digitizing probe. All the data information is saved into a specified file format. Two sets of data, one from CT image and the other from real time digitized data, are read into the optimization program developed at the University of Florida. The 6-D fitting parameters are then calculated to direct phantom movement for realignment. Verification and error analysis can be performed since the phantom was translated or rotated with

known values. Table 7-8 shows the difference of the calculated answers with the true known values. The calculated results show that high precision of 2mm treatment vector error can be achieved if the selection procedure of 3-D points is well performed. The error terms are generated from several factors, which are the inadequate selection of phantom CT contour line and the real time digitized contour line, the inadequate selection of the phantom anatomical points and the digitized anatomical points during experimental procedures. However, the spline fitting and the weighting process in the data manipulation stage tend to minimize the errors introduced from both the image contouring procedure and the real time digitized contouring procedure.

The experimental procedures described in this chapter provide for the logical examination of the system performance. First, the mathematical algorithms provide the necessary interface to calculate the 6-D fitting parameters. Four different algorithms are compared and verified for the most reasonable accurate mathematical outputs. These results deliver the information for the best fit for phantom repositioning.

Secondly, from the image-image correlation, the interior anatomical structures can be identified and correlated by visual and geometrical calculation. This process apparently is limited by the finite resolution of the CT pixel size. However, the correlation is easily observed as the image is translated and rotated to the new calculated location.

Thirdly, high precision real time digitized data correlation provides a tool that shows repositioning error can be minimized. The subsequent fractionated radiation treatment will contain smaller input error terms. The total system error can also be

Table 7-8: Error Analysis of the CT phantom image data and the digitized 3-D data points. A least-square matching is assumed in the object function and the calculated results are displayed.

	Experimental Conditions		
Average results of all the algorithms	No movement	10° rotation on Z-axis	30 cm movement on X and Y axes, 10° on Z axis
X (mm) Y (mm) Z (mm)	(0.626) (-0.495) (-0.823)	(0.775) (-0.759) (-1.804)	(31.527) (28.962) (1.154)
α (degrees) β (degrees) γ (degrees)	(-1.40) (1.57) (-2.32)	(10.31) (-1.85) (0.35)	(10.91) (0.38) (-3.15)
Error (Vector, mm)	(1.146)	(2.105)	(2.177)

reduced because the input data sets are the most accurate and reliable in obtaining the 3-D spacial coordinates.

Fourthly, the correlation of phantom image data and real time data provides a thorough check of the system performance. The phantom can be considered as a real patient who received CT or MR scans. The contour midline and anatomical structures are extracted to represent the patient rigid body information. This information has a relative geometrical connection with respect to the target isocenter. If the patient was set up in the initial treatment position, the same contour line and anatomical structures are taken again by the 3-D real time camera system. The calculation provides a convenient tool in tracing patient movement and treatment location a the time of setup. Therefore, the total accuracy of the data manipulation process can be estimated with this study. The treatment accuracy is the summation of this accuracy term in phantom tests plus the potential patient movement in the treatment procedure.

CHAPTER 8
THE HARDWARE SYSTEMS AND THE VERIFICATION PROCEDURES

System Characteristics

The purpose of building the fractionated stereotactic radiotherapy system is to collect patient information in a fast or real time manner and be able to reposition patient back to the original treatment planning coordinates. The information can be fed into a computer and the calculation of 6-D fitting parameters then becomes possible. Using the acquired transformation matrix from a current patient setup, the patient repositioning procedure is executed before the radiation treatment. Furthermore, the patient treatment position can also be monitored and verified during each treatment segment in fractionated stereotactic radiotherapy. The patient treatment location is adjusted as necessary before each treatment starts. The precision of treatment delivery can be confirmed and best treatment outcome can be achieved. The conceptual design process can be sketched as in Fig. 8-1.

High Precision Camera System

The Northern Digital™ 3-D camera system has its own design characteristics. Unlike systems that employ video based cameras and pattern recognition technology, this camera system uses three 1-D sensors and a cylindrical lens to pick up infrared LED signals for their specific frequency responses. Each sensor in the camera contains a Charge-Coupled-Device (CCD) that is divided into 2048 different pixels. The sensor has a field of view (FOV) at 35 degrees and a useful depth of field (DOF) from 1 m to 8 m.

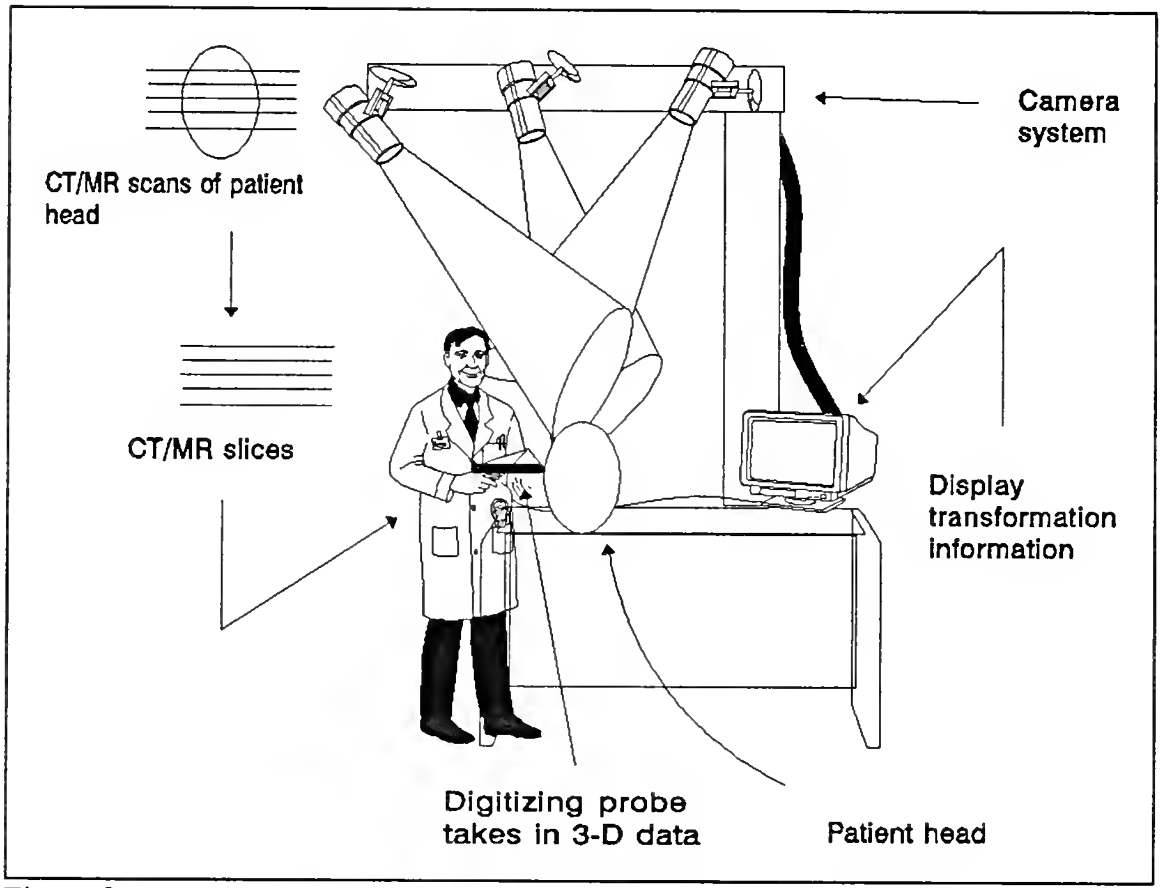

CT/MR scans of patient head
CT/MR slices
Camera system
Display transformation information
Digitizing probe takes in 3-D data
Patient head

Figure 8-1: Conceptual design of the digitizing and correlation process

Accuracy of the sensor is ± 1 part in 100,000 with the resolution of better than ± 1 part in 250,000. The lens system is based on a custom, multi-element, anamorphic design. This lens system stretches the image of a point in space into a tightly focused line which is projected orthogonal to the CCD linear array. Therefore, the lens is capable of producing a sharp focused image across the 35 degree FOV and for a DOF from 1 m to infinity without requiring changes in focus. Since the lens system does not require focusing, the lens elements and CCD array are mounted in a rigid unit and can be characterized using a unique mathematical model. A total of 19 parameters are determined to characterize features such as the focal length, the symmetrical and asymmetrical lens distortion of each sensor. Since these parameters are fixed, they have to be calibrated only once.

The calibration procedures were done at the factory to determine the interior parameters of each of the 1-D sensors and their respective position and orientation. This process requires the movement of the infrared LED markers through a large number of accurately known 3-D positions and the corresponding averaged centroid readings are recorded. The next step is to minimize the difference between what the individual 1-D sensors measure and what the mathematical model using these parameters predicts the sensors should be. This process is performed using a large scale, non-linear least squares algorithm with typically more than 60 unknowns and greater than 35,000 equations. The calibration provides the absolute coordinate system of the camera box and it will be fixed if the structure of the camera box is not damaged.

The LED marker emits infrared light that falls on each sensor of each camera at a different location. The centroid on the CCD array represents the physical location on the sensor and is used in the 3-D reconstruction process. This camera system uses a dynamic feedback mechanism to increase or decrease the amplification of the infrared signal received by the sensors. The mechanism is designed to maintain a constant signal level, regardless of the actual power received during the signal collection process. The power will vary depending on the distance of the marker from the camera, the angle from the marker to the sensor and the power of the marker. The infrared light emitted by a marker is focussed by the camera lens system into a tight line which falls across the sensor orthogonally. The sensor collects the light energy of the 2048 pixels separately.

If the amount of energy collected reaches the specific threshold, a characteristic waveform is drawn to represent response of each pixel output. A weighted centroid algorithm is applied to the pixels in order to determine the raw centroid value. A typical waveform of the CCD response can be observed in Fig. 8-2. Since the camera sensor is a linear device, the centroid value of the CCD array describes a plane in which the marker is located. By using multiple sensors to find the intersection of planes which were generated from linear CCD arrays, the 3-D coordinates of the marker in the reference camera coordinate system can be reconstructed. In order to determine the intersection of the three planes indicated by the centroid value of the sensor array, two of these planes must be oriented at right angles to the third one. The layout of the intersection planes of the camera system is as Fig. 8-3. The camera structure is formed by mounting the middle sensor vertically (showing a horizontal plane) and the other sensors are mounted horizontally (showing two vertical planes). The location of P in Fig. 8-3 is easily calculated from the triangular geometry and resolution of the marker position is defined from the linear resolution of the CCD array.

This camera system provides a useful tool for retrieving the 3-D coordinate information for high precision fractionated treatment purposes. In order to acquire very high precision treatment coordinates, the Northern Digital OPTOTRAK™ motion measurement system is utilized to perform the front end task in this project. This system has the following characteristics:

(1) It has a pre-calibrated 3-D coordinate system which eliminates user calibration process. This means that there will be only a simple transformation while moving the

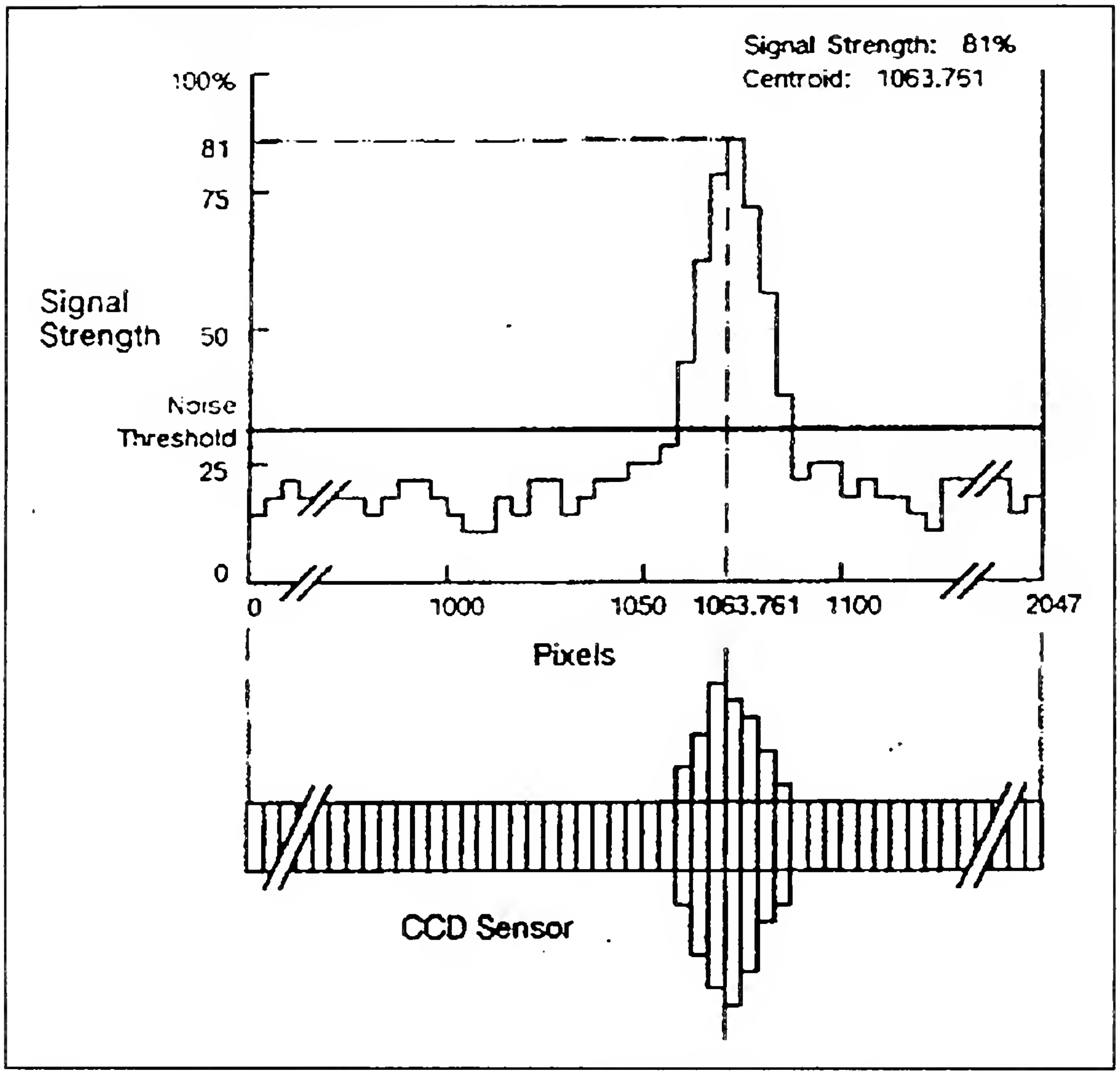


Figure 8-2: A characteristic waveform which illustrates the centroid of the infrared LED response to the CCD array. Noise threshold is also displayed. (Courtesy of Northern Digital™)

camera coordinate system to the LINAC isocenter coordinates. Once the isocenter is calibrated with respect to the camera system, the absolute coordinate system of the patient treatment location is can be established.

(2) The resolution is high. This camera system can resolve 0.01 mm at a distance of 2.5 m from the camera origin. It has 0.1 mm accuracy in the X and Y coordinates of the

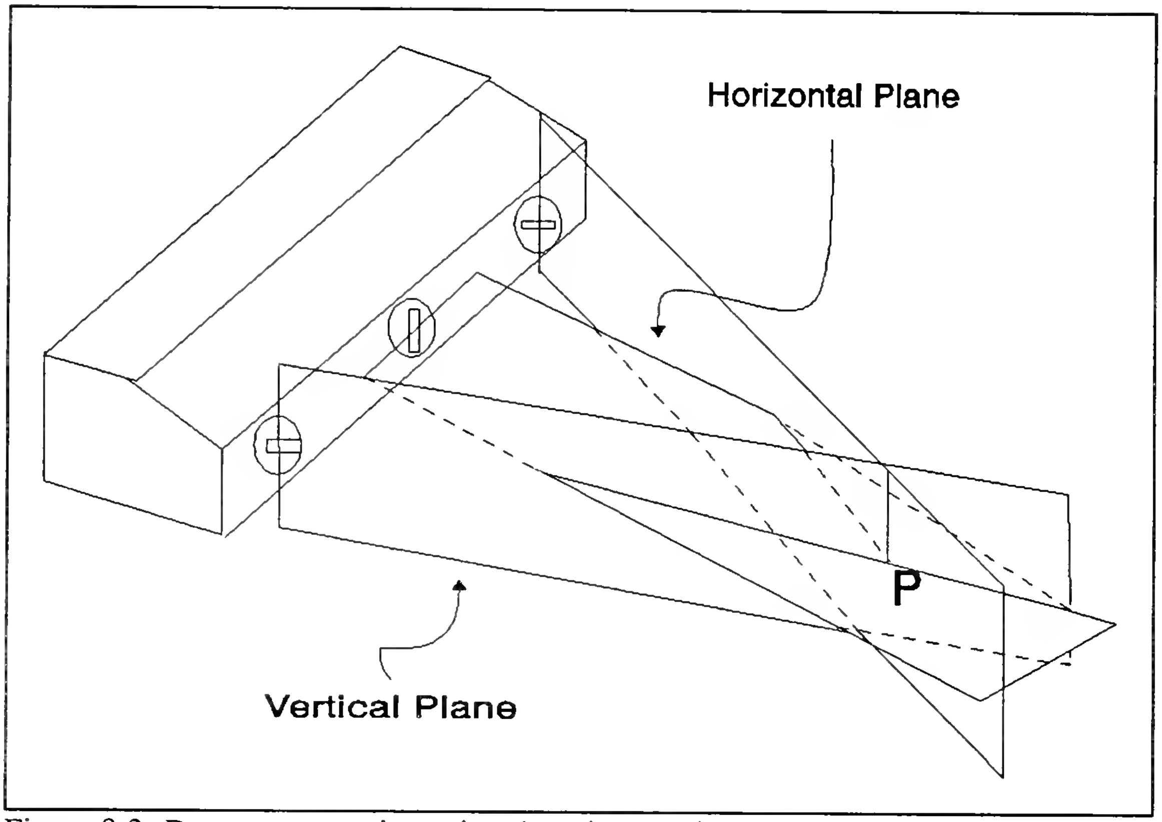


Figure 8-3: Data reconstruction using three intersecting planes of a CCD camera system

camera, 0.15 mm accuracy in 2.5 m depth. Per Northern Digital's documentation on camera resolution, this data may be acquired from a single measurement of a moving 8 mm target within a 1 m active area.

(3) The standard sampling rate is 3,500 markers per second. The large field of view (1 m X 1 m at 2 m distance and 2 m X 2.5 m at 4 m distance) can properly cover the patient target volume.

(4) The 3-D data are available either in real time or in a file format which can be retrieved for later examination. The infrared LED markers provide a large signal-to-noise ratio which can discriminate from other unnecessary signals that might contaminate

the data collection process.

The Real Time Digitizing Probe

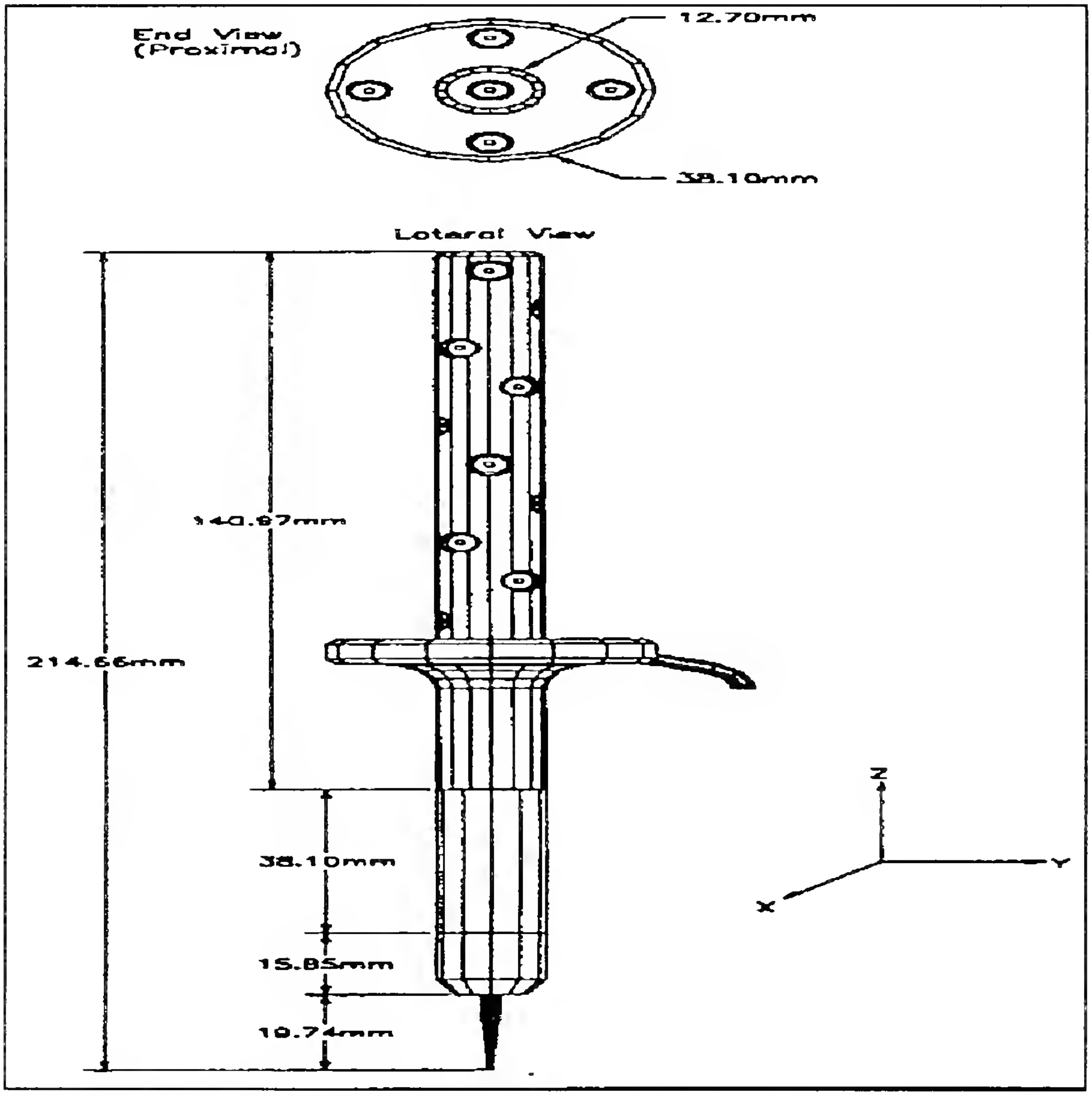


Figure 8-4: Lateral and end views of the 25 infrared LED markers probe. (Courtesy of Northern Digital™)

The digitizing probe provided by Northern Digital™ is used for defining the patient's surface information for the calculation of the 6-D fitting parameters (Fig. 8-4). A local coordinate system is established for the probe to process the transformation function from the camera coordinate system. The "rigid body" coordinate system of the digitizing probe provides important information in coordinate transformations by collecting patient anatomical contours or points. In simple terms, this rigid body is an object on which the infrared LED markers are fixed so that there is no relative movement between these markers and the digitizing probe itself. If a local origin and frame of reference are defined for the probe body and the position of each marker is measured relative to this frame of reference, special processing techniques may be used in order to provide information about the whole probe structure instead of the individual markers. The digitizing probe is simply a rigid body designed such that, using these processing techniques, a local coordinate is generated to describe the probe tip location. The local frame of reference for the digitizing probe is defined so that the origin is coincident with the probe's end-tip (Fig. 8-5). The local coordinate system is oriented so that the body of the probe lies along the Z-axis. Not only are the three positional coordinates of the probe end-tip represented, but also the three orientations of the coordinate system can be shown to describe the spatial location of the probe.

Calibration of the local probe coordinate system in order to match the LINAC treatment coordinate system is one of the critical steps in the patient realignment process. The purpose of this coordinate mapping procedure is to collect the 3-D information which represents the radiation delivery coordinates, not the camera coordinates. Therefore, correlation between the images data sets and real time digitized data sets becomes possible because the coordinate systems are the same - i.e., all refer to the LINAC treatment coordinate system. For example, suppose we wanted to measure a point on

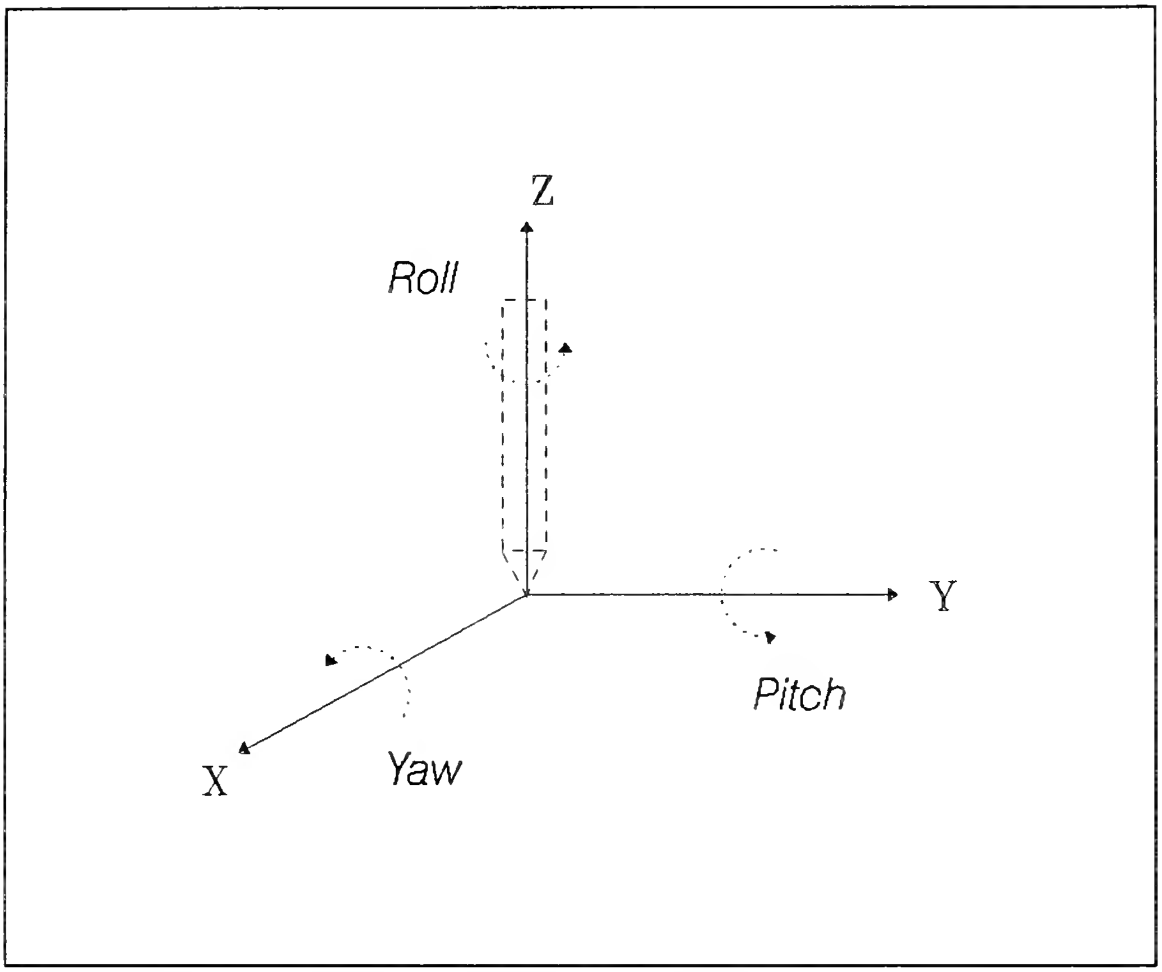


Figure 8-5: Rigid body local reference frame of the probe

the chin relative to the tumor, it would be more useful to measure this point relative to the isocenter of the machine (the local coordinate system) rather than to the calibrated camera coordinate system (the global coordinate system). The calibration procedure can be described as follows:

(1) Establish a coordinate system which correlates to the LINAC treatment coordinates. The origin of this coordinate system is correspondent to the machine isocenter. Three orthogonal axes should be exactly aligned with respect to the gantry movement, i.e., the beam delivery axes should overlap with the coordinate axes.

(2) Digitize the patient surface in both the contour midline format and in single anatomical point of 3-D data format. Once the necessary data are collected, a data processing and weighting procedure is performed to generate the proper data format for the transformation matrix calculation.

(3) Then the transformation matrix will be calculated. 6-D fitting parameters are calculated to realign the patient treatment position.

(4) Set up the calculated patient treatment location and again digitize contour midline and the external fiducial points. The correlation can be studied to analyze the new patient treatment location. This analysis shows the range of precision in the fractionated treatment of patients.

The Patient Immobilization System

One of the key parameters to guarantee a high precision fractionated treatment is to create a good patient immobilization device. Fortunately, with the head and neck region, an immobilization device can be more compact than for other body parts. This immobilization device allows the patient to be moved to any position and does not interfere with the stereotactic treatment arcs. Comfortable supporting material is required for the patient during each arc therapy. Although the treatment time is not long (about 30 minutes), an adequate resting device is necessary to hold the patient's head. Surface pressure is also necessary to hold the patient in position. A pressure pad is also a good choice.

The Patient Bite Plate Verification System

The patient final treatment location has to be verified with a reference device that provides information for the physician's confidence and for the legal records. Portal films can be taken to verify the target location. For 3-D verification information, two orthogonal films may be taken to reconstruct the necessary treatment location. For quality assurance purposes, the target location can be traced in a experimental setup. However, during the treatment procedure, the verification technique becomes difficult. The patient bite plate system is usually utilized to immobilize the patient and recover the treatment coordinates. Therefore, the patient target geometry should be correlated to the bite plate system. The bite plate system can be digitized during the treatment process and the correct treatment coordinates can be retrieved and compared.

From an anatomical point of view, the upper jaw provides a good bony relationship with the intracranial structures. Therefore, a bite plate system which employs a dental impression of the upper jaw carries good geometrical information for the fractionated treatment. This bite plate system also provides the external references for the treatment. Three lucite spheres are used to form a simple rigid body which carries the target information in the scanned images. Those spheres are again digitized with the LED probe to verify the relative location of the treatment coordinates. If the location of these spheres is the same as that in the scanned image set, the patient is confirmed at the correct coordinates and the treatment is ready to start.

The Verification Procedure for Fractionated Treatment

After the patient was set up correctly in the calculated and dialed coordinates, it is important to verify the location of the tumor isocenter with respect to the machine isocenter. The following process describes how the experimental verification procedures are accomplished and how isocenter offset will be studied.

For the experimental setup, a phantom with an identifiable contour line and external fiducial markers is scanned with a BRW localizer attached. The advantage of this test is that all the defined coordinates can be related back to the BRW coordinate system, which is the most accurate reference coordinate system in the clinical environment. Once the necessary information is incorporated into the treatment planning process, then the phantom midline contour, external fiducial markers, target location and other anatomical data are all known in the BRW coordinates. These basic parameters are considered as the reference data set for further calculation.

The 3-D digitizing camera system was calibrated to the isocenter of the Philips™ SRS 200 radiosurgery stand developed at the University of Florida. The real time digitizing probe captures the patient contour and anatomical information that corresponds to the diagnostic image data sets. Once the necessary data sets are acquired, the calculation can start and the required transformation parameters to reposition the phantom are displayed. The phantom is then moved to the desired location, according to the calculated results. Target verification is performed.

The LINAC treatment coordinate with respect to the camera system can be established by a tooling ball device which is attached to the digitizing probe. A proper description of the treatment coordinate system is required. After calibration of the treatment coordinates, the phantom is set to the coordinates which were calculated from the radiosurgery treatment planning software. The isocenter is dialed to the desired values and the probe then takes in phantom information again. The phantom target location consisting of an aluminum ball which correlates to the isocenter position is filmed. When the corresponding coordinates of three external fiducial markers are again digitized or the LED markers are captured, the treatment coordinates can be calculated and validated. This technique provides solid proof that the target location is correlated to the radiation treatment field. After extensive testing on a phantom base, the following results are reported.

In order to acquire high precision phantom test environment the BRW coordinate system is again utilized to verify the calculation results. A phantom was built with tungsten ball and nylon spheres to imitate the tumor location and bite plate system. Extensive data analysis procedures are performed and the results are reported. Table 7-1 show the experimental results with LED markers attached to nylon balls. The nylon balls are separated into two groups, lower group has more restrictive triangular shape and the upper group is more widely separated. The purpose of this kind of arrange is to study the possibility of translation error which might be introduced from the geometrical shape of this bite plate system. A total of twenty frames with a collection time of one second is operated in data acquiring procedures. The average coordinates are implemented instead of using the single frame of data set. The error bar on the total average data should be minimized.

In order to test the working concept of digitized contour line and the anatomical points the same phantom is again utilized. A total of five sets contour and anatomical data points are taken with known BRW coordinate settings. After applying the spline fitting and data weighting procedures which were described in previous chapter, the repeatability of the contour line and anatomical concept is again verified. The repeatability results are reported in the following Table 7-2.

The experimental environment used in testing the LED markers has been applied again to evaluate the concept of matching contour line and anatomical points. A total number of ten combinations has been summarized in the following Table 7-3.

By simply correlating the anatomical points, the transformation parameters can also be calculated. Four average digitized data sets which obtained from the repeatability study in Table 7-2 were used to calculate the error terms. Table 7-4 is then reported while using only the anatomical points for correlation. Again the CT contour line and anatomical points which acquired from the IMGLOC™ program are saved and exported to the PC as two separate files. The data manipulating process is performed to organized a complete data set from the CT image to the digitized data set. The same program is utilized to evaluate the best match of the CT and digitized data, which delivers information about how well these two can be correlated. Simply from transferring the phantom to the BRW stand should show the optimum solution of no translation and rotation. However, our calculation shows that 2.5 mm vertical shift and 2.3° lateral rotation are necessary to best fit the CT and digitized data sets.

It can be concluded that since CT is a relatively high noisy data set, the error term is from the data collection process of CT contour line and anatomical points. Using proper weighting function to filter out the noisy system error might be another approach to eliminate the error in CT data collection procedures.

Table 7-1: Phantom position error analysis with the LED data on BRW™ stand

Experimental number	Experimental conditions (All in BRW™ coordinate system)	Vector error in mm of lower group	Vector error in mm of upper group
(1)	(0, 10, 0)	0.40	0.36
(2)	(0, 10, 10)	0.10	0.21
(3)	(10, 10, 10)	0.19	0.23
(4)	(-27.35, 8.54, 41.72)*	0.26	0.64
(5)	(-17.35, -1.46, 31.72)**	0.53	0.32
(6)	Rotate +10° around AP-axis from (5)	0.27	0.35
(7)	Rotate +20° around AP-axis from (5)	0.96	0.24
(8)	Rotate -10° around AP-axis from (5)	0.45	0.05
(9)	Rotate -20° around AP-axis from (5)	0.09	0.27

Experimental number	Experimental conditions (All in BRW™ coordinate system)	Vector error in mm of lower group	Vector error in mm of upper group
(10)	Rotate +5° around AP-axis from (5)	0.23	0.18
(11)	Rotate -5° around AP-axis from (5)	0.87	0.16
Average error		0.40	0.27
Standard deviation		0.29	0.15

*: The target coordinate obtained from the Angio localization procedures

**: Shift 10 mm relative to the target point from the Angio localization procedures

Table 7-2: Repeatability study of the contour line and anatomical points registration technique

Experimental number	Experimental conditions (BRW™)	Average error (mm)	Standard deviation
(1)	(-27.35, 8.54, 41.72)	0.51	0.36
(2)	(-17.43, -1.46, 31.72)	0.92	0.24
(3)	Rotate +10° around AP-axis of (2)	0.97	0.27
(4)	Move to (0,0,0) of (3)	0.70	0.03
(5)	Rotate back to 0° of (4)	0.35	0.13

Table 7-3: Testing of the optimization algorithms using contour line and anatomical points concept instead of using LED markers.

Experimental number	Experimental conditions (All in BRW™ coordinate system and relates to isocenter location)	Vector error in mm
(1)	Translate to (-20, 0, +30)	0.62
(2)	Repeat (1) with another experimental data set	1.20
(3)	Rotate +10° of (1)	1.92
(4)	Repeat (3) with another experimental data set	1.19
(5)	Move to (0,0,0) and maintain +10° rotation	1.29
(6)	Repeat (5) with another experimental data set	1.40
(7)	repeat (5) with third experimental data set	1.16

Experimental number	Experimental conditions (All in BRW™ coordinate system and relates to isocenter location)	Vector error in mm
(8)	Rotate back to 0° of (5)	1.95
(9)	Repeat (8) with another experimental data set	2.00
(10)	Repeat (9) but twice the weighting factor of the anatomical points	0.73
Average error		1.35
Standard deviation		0.56

Table 7-4: Error analysis of the repositioning with averaged digitized data set

Experimental number	Experimental conditions (BRW™)	Average error (mm)
(1)	(-27.35, 8.54, 41.72)	N/A
(2)	(-20.00, 0.00, 30.00)	0.97
(3)	Rotate +10° around AP-axis of (2)	0.32
(4)	Move to (0,0,0) of (3)	0.78
(5)	Rotate back to 0° of (4)	0.52
Average error		0.65
Standard deviation		0.29

Conclusion

The data verification procedures mentioned in this chapter provides a numerical expression of how the design concept and the system work. These experimental data sets can introduce the possibility of further investigation of the error sources terms. The weakest link occurs in defining the image data as the standard fractionated treatment information. This can be resolved by well defined contour line segment and anatomical points. External fiducial markers are also the possible choice to define a good image data set. This 3-D camera system does provide a very accurate tool in acquiring 3-D data information once is well calibrated. The optimization program also provides a fast and accurate method to analyze the data and delivers transformation parameters. If the system noise can be further reduced, the error in the real time correlation calculation can be improved.

CHAPTER 9
CONCLUSION

Conclusion

High precision treatment of fractionated stereotactic radiotherapy has provided a very useful and accurate tool to achieve the goal of radiation therapy - i.e., delivery of maximum radiation dose to the tumor and minimum dose to normal or critical tissues. This work has suggested and implemented the conceptual framework and experimental studies of Fractionated Stereotactic Radiotherapy at the University of Florida. Based upon the designed software and hardware system, high precision fractionated treatment of intracranial tumors may become possible. Phantom studies and patient image studies have been performed to evaluate the treatment accuracy. The results are promising and the clinical implementation will be engaged in the near future. A sophisticated camera system provides the necessary accuracy in determining the 3-D patient position. Point matching algorithms as well as surface matching algorithms provide useful mathematical derivation for repositioning the patient in subsequent fractionated treatment location. With proper immobilization techniques the phantom movement and treatment precision have experimentally proved that the total inaccuracy is acceptable for the optimum treatment with high precision. Therefore, the mathematical algorithms and future treatment techniques provide a reliable treatment delivery system in the application of a more precise fractionated treatment system. Although at this point in time this equipment setup has not produced sum millimeter treatment precision, we are only at the beginning of our experimental work.

The standard reference data set used to describe the patient's rigid body information which was formed from the scanned images is quite important. The finite resolution of the image pixel size tends to zigzag along the patient contour line. Therefore, a sorting and a cubic spline fitting function are applied to smooth the patient contour acquired from the image. Other specified anatomical points are also selected to provide rigid body information with the contour line for calculation purposes. This process tends to correct human-machine interference errors introduced from the registration of a reference data base for treatment planning.

The mathematical algorithms in calculating the 6-D fitting parameters have been extensively studied to obtain numerically stability. For the least-squared object function to correlate two 3-D data sets, these optimization algorithms reach close agreement in approaching the best fit answers. With respect to the SVD algorithm, optimization algorithms actually are more easily understandable and programmable. Speed and stability of these optimization algorithms are another advantage for the calculation process on a personal computer. Those programs are considered user friendly and can be compared or verified during the data manipulation procedures.

In conclusion, the mathematical functions developed and the methodology created in this dissertation provide a direct tool to solve the patient repositioning problem during the fractionated radiation therapy treatment. Experiments show precision within 1 mm using phantom anatomical structures in repositioning without using an invasive immobilization device, while offering ten times the precision when compared to routine radiation therapy procedures. This work has presented an effective system to approach fractionated radiation therapy patient repositioning and monitoring problems.

Future Work

As the University of Florida Fractionated Stereotactic Radiotherapy is applied in daily tumor treatment, the collection and analysis of clinical patient data will be studied in the future. Long term followup and comparison are necessary to evaluate the treatment outcome. Since the Fractionated Stereotactic Radiotherapy incorporates the biological advantages of this precision treatment, the following tasks are suggested for future progress.

(1) Time-dose-fraction study:

The time-dose-fraction scheme used and the volume irradiated will vary depending on the tumor size, location and cell type. The proposed treatment scheme at this point is the same as that in the routine radiation treatment of the other body parts, which is five fractions a week. Fractionation increases the cellular depopulation of the tumor for a given total dose because of re-oxygenation. At the same time, fractionation reduces the damage to the critical late responding normal tissues. Therefore, the sensitivity correlation between the tumor and late responding tissue with respect to the fractionation schemes is very important. An study should be performed by varying the fractionated treatment schemes and the followup should be analyzed to optimize the fractionated treatment.

(2) Fractionated Stereotactic Radiotherapy on other body sites:

The same technique in fractionated stereotactic radiotherapy can be potentially applied in treating other body sites. Because treatment portals can be designed with non-coplanar arcs, critical organs or areas next to tumor volumes will have better sparing effect. With existing immobilization devices and repositioning techniques developed in treatment of the head area, the potential usage of fractionated stereotactic radiotherapy could create a new treatment protocol in the future.

(3) Real time frameless stereotactic surgery in the Operating Room (OR)

There has been a remarkable increase in the use of CT and MR imaging to guide the intracranial surgical procedure. A typical procedure is that the operation starts with the proper registration of the preoperative images using external fiducial markers. Therefore, the surgical field is then correlated to the image set in a specific coordinate system. These fiducial markers are either applied directly to the patient head or indirectly attached to the rigid frame which contains its own coordinate system, for example, the CRW™ coordinate system. The fiducial markers require image data sets immediately prior to surgery which introduces inconvenience in the operational procedures. Disadvantages of the framed stereotactic method are the restricted usages and the interferences between the surgical probe and the frame during the operation.

Frameless stereotactic surgery in conjunction with 3-D image presentation allows for accurate planning and performance of a variety of neurosurgical or the other surgical procedures. There will be no obstacles in performing the real time intracranial procedure. Since we have already collected experience with the high resolution camera system, the frameless stereotactic neurosurgery may become the trend for defining surgical coordinates and introduces an easier method to correlate different image modalities. The frameless stereotactic neurosurgery should satisfy the following requirements:

(a) The object which will be operated on and its image sets have to be displayed in real time. The camera coordinate and the image coordinate should correlate in a common domain, i.e., they are well calibrated to a common origin in a specified coordinate system.

(b) The position of the surgical instrument has to be simultaneously faded into the display of the image data sets.

In real time neurosurgery, the patient-image coordinate transformation was found by retrospectively registering a camera system model of the patient's anatomy with a CT or MR derived image model. This procedure will guarantee the routine image data sets can be used to perform the surgical procedure, no special scan is required prior to the surgery.

(4) Virtual Reality studies in neurosurgery and radiation oncology:

Virtual Reality (VR) has become a popular technique in medical uses involving using microscopic TV cameras or stereo fiber optics to perform microsurgery with minimal incisions. A high resolution display enables the surgeon to view the blood vessels and nerves with refined 3-D details. A headband mounted display provides full view of a patient's operation and examination sites. The 3-D camera system provides a tool for entering the coordinates in a virtual reality world. Therefore, a surgeon can actually enter a patient "virtually" and view the diseased area from within.

The matrix transformation applied in this dissertation supplies a necessary interface to transform the real world coordinate to the image or real time data set coordinate system. The patient data bases in either neurosurgery or radiation oncology can be incorporated into the "virtual environment" and be retrieved from any viewing angle or depth. Users can select the desired 3-D coordinates and manipulate the large volume of information with high speed display. Hopefully the surgery or the treatment coordinates can also be designed to perform the task on an image basis without patient interruption. A feasibility study of the application and the creation of preliminary virtual environment could be pursued in the near future.

APPENDIX A
COORDINATE TRANSFORMATION

Mapping of two sets of 3-D data points consists of translation and rotation manipulation. The transformation between two coordinate systems can be interpreted as operators which translate and rotate vectors in 3-D space. They can be characterized as follows in separate items.

Translational operators

The translation moves a point in space a finite distance along a given vector direction. Translating a point in space is accomplished with the same mathematics as mapping the point to a second coordinate system. Moving a vector is the same as inversely moving the reference coordinate system. The distinction is as simple as this: when a vector is moved "forward" relative to a coordinate system, we may consider either that the vector moved "forward" or that the coordinate system moved "backward". The mathematics involved is identical, only our view of the situation is different.

Fig. A-1 indicates pictorially how a vector ${}^{A}P_1$ is translated by a vector ${}^{A}Q$. The ${}^{A}Q$ gives the information needed to perform the translation which is ${}^{A}P_2 = {}^{A}P_1 + {}^{A}Q$. The translational operator can also be described in Equation A-1. This homogeneous transformation matrix represents the linear shifts in 3-D coordinates. q_x, q_y, q_z represent the linear movements along three orthogonal axes.

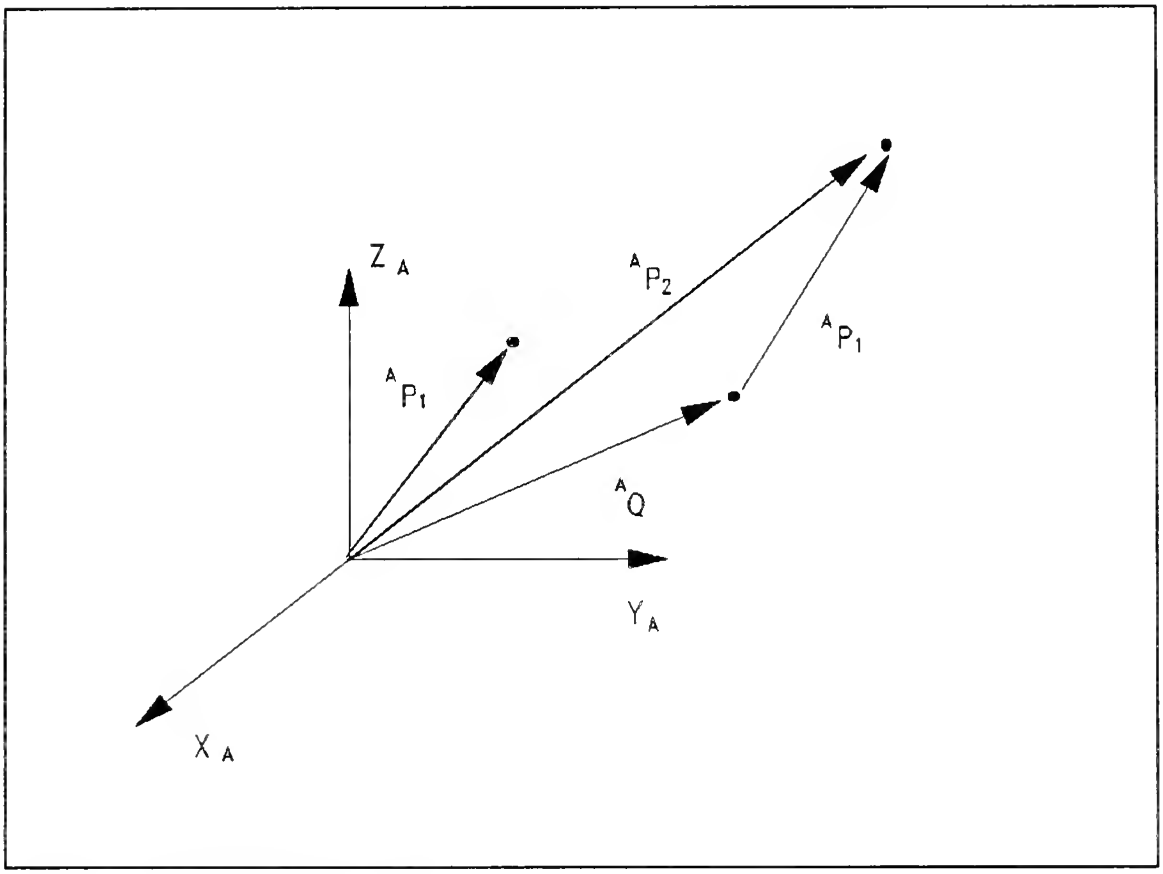


Figure A-1: Translation operator

$$\boldsymbol{TRANS} = \begin{vmatrix} 0 & 0 & 0 & q_x \\ 0 & 0 & 0 & q_y \\ 0 & 0 & 0 & q_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad \text{(A-1)}$$

<u>Rotational operators</u>

An interpretation of a rotation matrix is a rotational operator which operates on a vector $^AP_1$ and changes that vector to a new vector, $^AP_2$, by means of a rotation, R. Usually, when a rotation matrix is shown as an operator it is viewed as relating one coordinate system. The rotation matrix which rotates vectors through some rotation, R, is the same as the rotation matrix which describes a coordinate system rotated by R relative to the reference frame. For instance, the rotational operator in the Z-axis rotation in the 3-D space can be described in Equation A-2 and the geometrical representation is shown in Fig. A-2.

$$\boldsymbol{ROT}(Z,\theta) = \begin{vmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad \text{(A-2)}$$

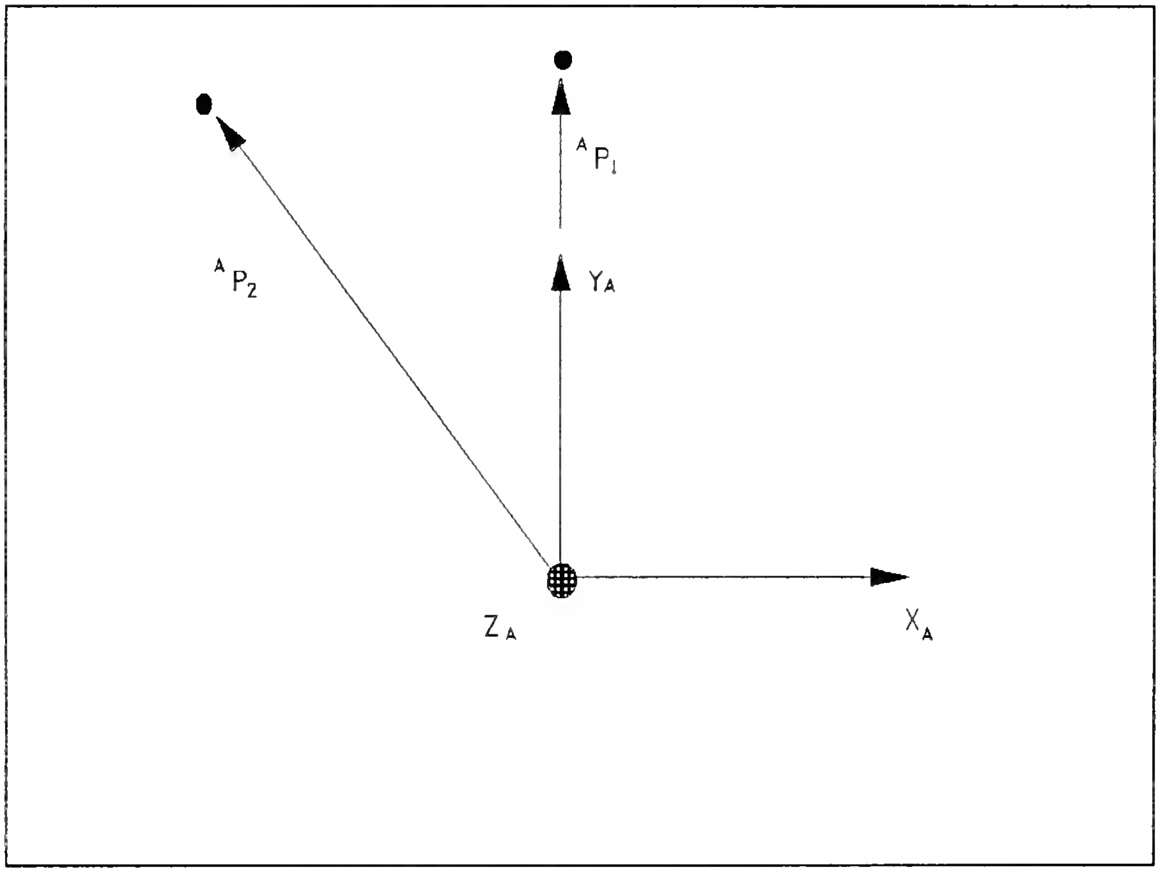


Figure A-2: Rotation operator (Z axis points out from the paper)

Compound transformation

The multiple operations of translations and rotations form a functionally complete set of coordinate transformation. The compound transformation will bring certain points or vectors to the specified position and orientation. Homogeneous matrix multiplication can transform a series of translation and rotations into one simple matrix format, in which all the necessary information can be obtained. Since 3-D translation is an easy operation to move the points or vectors, therefore we only explain the multiple rotations. Assume that two coordinate systems can be related as follows:

Start with frame {B} (patient's head frame) coincident with a known frame {A} (LINAC frame). First rotate {B} about Z_B by an angle α, then rotate about Y_B by an angle β, and then rotate about X_B by an angle γ.

Note that in the matrix multiplication, each rotation is performed about the rotated axis of {B}, rather than the fixed reference coordinate system. Such a set three rotations are so called *Euler angles.* Each rotation takes place about an axis whose location depends upon the preceding rotation. Because the rotation occurs about the Z, Y and X axes, it is also recognized as *Z-Y-X Euler angles.* Fig. A-3 shows the axes of {B} after each Euler angle rotation is applied. An additional "prime" sign is added after each corresponding rotation to identify the rotating frame. Suppose we wish to find a rotation matrix as a function of the Z-Y-X Euler angles, $R_{ZYX}(\alpha,\beta,\gamma)$. Because the transformation of a coordinate system follows an opposite order of matrix multiplication than the transformation of space vectors, the final description of any vector ^{A}P is thus seen to be
The solution characterizes the three Euler angles. Therefore, Euler angles α, β, and γ can be calculated from the information contained inside the matrix elements.

Since rotations are caused by matrix multiplication, 3-D rotations are non-

$$R_{ZYX}(\alpha,\beta,\gamma) = \boldsymbol{ROT}(Z,\alpha)\ \boldsymbol{ROT}(Y,\beta)\ \boldsymbol{ROT}(X,\theta)$$
$$= \begin{vmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$
$$= \begin{vmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma-\sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma & 0 \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma-\cos\alpha\sin\gamma & 0 \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

(A-3)

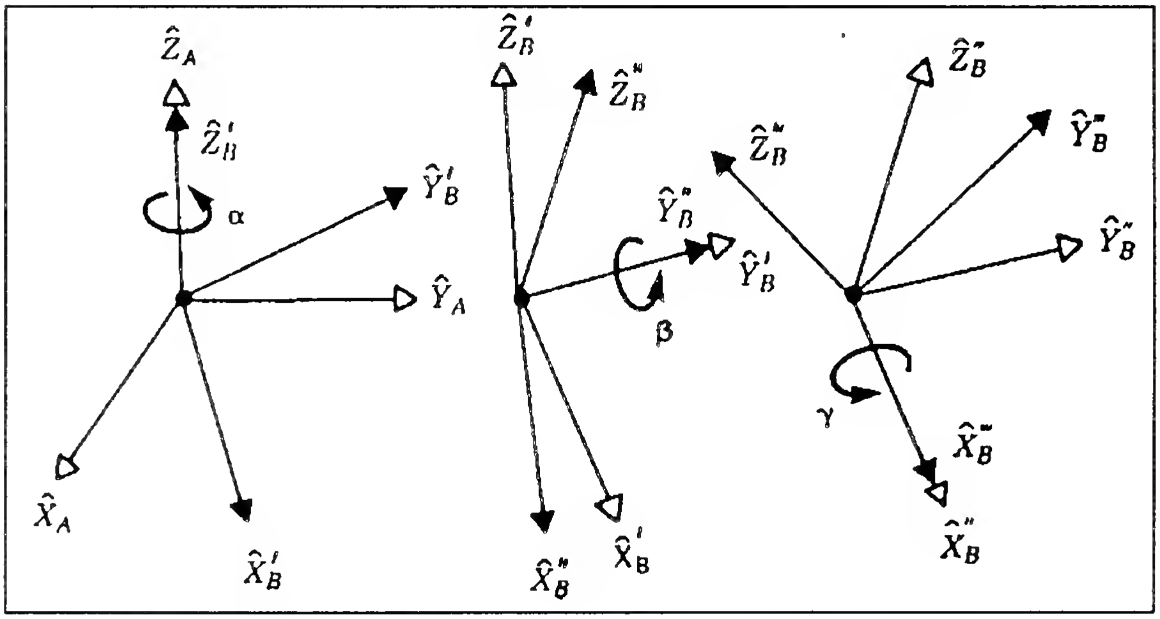


Figure A-3: Z-Y-X Euler angles

commutative; i.e., the order of multiplication will affect the final results. In order to prove this, consider a rotation about the x-axis followed by an equal rotation about the y-axis with an angle of θ. We have the following relationship in equation A-4 and equation A-5: From the above equations we can easily verify the matrix elements are not correspondent. Therefore, the sequence of rotations should be carefully administrated while performing a series of 3-D rotations.

$$\boldsymbol{T} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$
$$= \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ \sin^2\theta & \cos\theta & -\cos\theta\sin\theta & 0 \\ -\cos\theta\sin\theta & \sin\theta & \cos^2\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad \text{(A-4)}$$

$$\boldsymbol{T} = \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$
$$= \begin{vmatrix} \cos\theta & \sin^2\theta & \cos\theta\sin\theta & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ -\sin\theta & \cos\theta\sin\theta & \cos^2\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad \text{(A-5)}$$

If the Euler rotation is not specified on the same rotational origin for each axis, the new definition for each rotation on different axis is required. If the Euler rotation occurs as the same sequence with respect to the Fig. A-3 and rotates around different origins, the following matrix multiplication is then applied:

(1) Before rotation around the Z axis, the origin is translated with known coordinates of (T1,T2,T3).

(2) Before rotation around the Y axis, the origin is translated with known coordinates of (T4,T5,T6).

(3) Before rotation around the X axis, the origin is translated with known coordinates of (T7,T8,T9).

Therefore the following equation is encountered:

$$\begin{aligned} &\boldsymbol{TRANSFORMATION}_{ZYX}(\alpha,\beta,\gamma) = \\ &\boldsymbol{ROT}(Z,\alpha)\,\boldsymbol{TRANS}(T7,T8,T9) \\ &\boldsymbol{ROT}(Y,\beta)\,\boldsymbol{TRANS}(T4,T5,T6) \\ &\boldsymbol{ROT}(X,\gamma)\,\boldsymbol{TRANS}(T1,T2,T3) \end{aligned}$$

$$= \begin{vmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0 & 0 & 0 & T7 \\ 0 & 0 & 0 & T8 \\ 0 & 0 & 0 & T9 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$\cdot \begin{vmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0 & 0 & 0 & T4 \\ 0 & 0 & 0 & T5 \\ 0 & 0 & 0 & T6 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad \text{(A-6)}$$

$$\circ \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 0 & 0 & 0 & T1 \\ 0 & 0 & 0 & T2 \\ 0 & 0 & 0 & T3 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$= \begin{vmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma-\sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma & K1 \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma-\cos\alpha\sin\gamma & K2 \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & K3 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

where $K1 = \cos\alpha\cos\beta(T1+T4) + \cos\alpha\sin\beta(T3+T6) + \cos\alpha(T7) - \sin\alpha(T2+T5+T8)$

$K2 = \sin\alpha\cos\beta(T1+T4) + \sin\alpha\sin\beta(T3+T6) + \sin\alpha(T7) - \cos\alpha(T2+T5+T8)$

$K3 = -\sin\beta(T1+T4) + \cos\beta(T3+T6) + T9$

The translation coordinates depend upon the physical device whose measured dimensions will provide the necessary information for T1 to T9. The rotational sequence is again quite important because of the non-commutative characteristics of the matrix multiplication. The assumptions for rotational sequence is defined in Z-Y-X axis order while the translational coordinates are the rotational origin on each axis. If changes were made or dimensions were altered, the re-computation is necessary for the calculation for 6-D fitting parameters.

APPENDIX B
COMPUTER PROGRAMS

The following pages consists of different listings of computer programs described in previous chapters. The programs (PROS, HOOKE, SIMPLEX, and ANNEAL) contain the mathematical analysis of the least-squared solution and optimization approach in calculating the translational and rotational fitting parameters. Random simulation of clinical data sets with Gaussian distribution (RANDOM) provides a tool for various study conditions. Pre-processing, spline fitting and linear weighting functions (FILTER, SPLINE, and WEIGHT) of CT/MR contour and anatomical data points as well as 3-D real time digitized data are also included. All code is written in ANSI C and compiled with the TURBO C++™ application software in personal computer.

```
/*************************************************************************/
/*      PORS.C        */
/* Procrustes point registration - read in lists of homologous points,
   form matrix of scalar products of all point pairs, do singular
   value decomposition to diagonalize yielding transformation which
   matches points in least squares sense.

   Edited by Jack Yang        */

/* This program is to rotate and translate the 2nd day data points to match
   the 1st dat data points. This situation is similar to change the 2nd
   digitized contour points to match the CT contour points, which is exactly
   what we need to do in the future */
/*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/* Definitions */
typedef float XYZPOINT[3];
typedef float **NRMATRIX;
typedef float *NRVECTOR;

 static float at,bt,ct;
#define PYTHAG(a,b) ((at=fabs(a)) > (bt=fabs(b)) ? \
(ct=bt/at,at*sqrt(1.0+ct*ct)) : (bt ? (ct=at/bt,bt*sqrt(1.0+ct*ct)): 0.0))
static float maxarg1,maxarg2;
#define MAX(a,b) (maxarg1=(a),maxarg2=(b),(maxarg1) > (maxarg2) ?\
        (maxarg1) : (maxarg2))
#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))

#define PI 3.141592654
#define RADIAN (PI / 180.0)
#define A_SIGN(a1,a2) (a2 < 0 ? -1.0 * fabs((double) a1) : a1)

/* FUNCTION PROTOTYPES DECLARATION */

float *vector(int nl, int nh);
float **matrix(int nrl, int nrh, int ncl, int nch);
void free_vector(float *v, int nl, int nh);
void free_matrix(float **m, int nrl, int nrh, int ncl, int nch);
void nrerror(char error_text[]);
```

```
char *GetaString(char *String, char *Prompt);
void svdcmp(float **a, int m, int n, float *w, float **v);
void solve_euler(float amat[3][3], float *x, float *y, float *z);
void rotY(float thet, float amat[3][3]);
void Mult3x3(float amat[3][3], float bmat[3][3], float cmat[3][3]);

/* TRANSPOSE OF A FUNCTION */
static void transpose(a, nl, na)
NRMATRIX a;
int nl, na;
{
  int i,j;
  float temp;

  for (i = nl; i <= na; i++)

     for (j = nl; j <= na; j++) {
        temp = a[i][j];
        a[i][j] = a[j][i];
        a[j][i] = temp;
     }

  return;
}

/* UNLOAD A MATRIX */
static void UnloadNRmatrix(a, nrl, nrh, ncl, nch, dest)
NRMATRIX a;
int nrl, nrh, ncl, nch;
float *dest;
{
  int i, j;
  float *fptr = dest;

  for (i = nrl; i <= nrh; i++)
    for (j = ncl; j <= nch; j++, fptr++) (*fptr) = a[i][j];
}

/* LOAD A MATRIX */
static NRMATRIX LoadNRmatrix(source, nr, nc)
float *source;
int nr, nc;
{
  int i, j;
  NRMATRIX temp;

  temp = matrix(1, nr, 1, nc);

  for (i = 1; i <= nr; i++)
    for (j = 1; j <= nc; j++, source++)  temp[i][j] = *source;
  return(temp);
}

/* CALCULATE THE TRACE OF A MATRIX */
static float NRtrace(a, nrl, nrh)
NRMATRIX a;
int nrl, nrh;
{
  int i;
  float sum = 0.0;

  for (i = nrl; i <= nrh; i++) sum += a[i][i];
  return(sum);
}

/* SUBTRACTION OF TWO MATRICES, subtracts b from a */
static NRMATRIX NRmatrixSubtract( a, b, nrl, nrh, ncl, nch)
NRMATRIX a, b;
int nrl, nrh, ncl, nch;
{
   int i,j,k;
   NRMATRIX temp;

   temp  = matrix(nrl, nrh, ncl, nch);

   for (i = nrl; i <= nrh; i++)
     for (j = ncl; j <= nch; j++) {
        temp[i][j] = a[i][j] - b[i][j];
     }
   return (temp);
}

/* PRODUCT OF TWO MATRICES */
/* multiplies a (na x nb) times b (nb x nc) returning c (na x nc) */
static NRMATRIX NRmatrixProduct( a, b, na, nb, nc)
NRMATRIX a, b;
int na, nb, nc;
{
  int i,j,k;
  NRMATRIX temp;

  temp  = matrix(1, na, 1, nc);

  for (i = 1; i <= na; i++)
    for (j = 1; j <= nc; j++) {
      temp[i][j] = 0;
      for (k = 1; k <= nb; temp[i][j] += a[i][k] * b[k][j], k++);
    }
  return (temp);
}

/* ALLOCATE TRANSPOSED MATRIX */
/* allocates and fills transpose of general NR matrix */
static NRMATRIX NRmatrixTranspose( a, nrl, nrh, ncl, nch)
NRMATRIX a;
int nrl, ncl, nrh, nch;
{
  int i,j,k;
  NRMATRIX temp;

  temp  = matrix(ncl, nch, nrl, nrh);

  for (i = nrl; i <= nrh; i++)
    for (j = ncl; j <= nch; j++) temp[j][i] = a[i][j];
  return (temp);
}

/* CALCULATE THE NORM OF A MATRIX (EITHER COLUMN OR ROW) */
/* calculates row or column norm of matrix */
static NRVECTOR NRrowColNorm( a, nrl, nrh, ncl, nch, rows)
NRMATRIX a;
int nrl, ncl, nrh, nch, rows;
{
  int i,j,k;
  NRVECTOR temp;

  if (rows) {
    temp  = vector(nrl, nrh);

    for (i = nrl; i <= nrh; i++){
      temp[i] = 0.0;
      for (j = ncl; j <= nch; j++) temp[i] += a[i][j] * a[i][j];
```

```
        temp[i] = (float) sqrt((double)temp[i]);
      }
    }
     else    {
      temp  = vector(ncl, nch);

      for (i = ncl; i <= nch; i++){

        temp[i] = 0.0;
        for (j = nrl; j <= nrh; j++) temp[i] += a[j][i] * a[j][i];
        temp[i] = (float) sqrt((double)temp[i]);
      }
     } /*    if (row)  */
   return (temp);
}

/* CALCULATE THE ROW SHIFT OF A MATRIX */
static void NRrowShift(mat, vec, nrl, nrh, ncl, nch, subtract)
NRMATRIX mat;
NRVECTOR vec;
int nrl, nrh, ncl, nch, subtract;
{
  int i, j;
  float factor;

  factor   = (subtract ? -1.0 : 1.0);

  for (i = nrl; i <= nrh; i++)
    for (j = ncl; j <= nch; j++) mat[i][j] += factor * vec[j];
  return;
}

/* CALCULATE THE COLUMN SHIFT OF A MATRIX */
static void NRcolShift(mat, vec, nrl, nrh, ncl, nch, subtract)
NRMATRIX mat;
NRVECTOR vec;
int nrl, nrh, ncl, nch, subtract;
{
  int i, j;
  float factor;

  factor   = (subtract ? -1.0 : 1.0);

  for (i = ncl; i <= nch; i++)
```

```
    for (j = nrl; j <= nrh; j++) mat[j][i] += factor * vec[j];
  return;
}

/* SCALE A MATRIX */
static void NRscale(mat, scale, nrl, nrh, ncl, nch)
NRMATRIX mat;
float scale;
int nrl, nrh, ncl, nch;
{
  int i, j;
  for (i = nrl; i <= nrh; i++)
    for (j = ncl; j <= nch; j++) mat[i][j] *= scale;
  return;
}

/* PRINT OUT THE CALCULATED MATRIX */
static void PrintNRmatrix(title, amat, il, ih, jl, jh, byrows)
char *title;
NRMATRIX amat;
int il, ih, jl, jh, byrows;
{
  int i,j;

  printf("%s", title);
  if (byrows)
  for (i = il; i <= ih; i++)  {
    for(j = jl; j <= jh; j++) printf("%.2f ", amat[i][j]);
    printf("\n");
  }
  else /*  by columns */
    for(j = jl; j <= jh; j++)  {
     for (i = il; i <= ih; i++) printf("%.2f ", amat[i][j]);
     printf("\n");
    }
}

/* DEFINE VECTOR */
float *vector(int nl, int nh)
/* Allocate a float vector with subscript range v[nl..nh] */
{
        float *v;

        v=(float *) malloc((nh-nl+1)*sizeof(float));
        if (!v) nrerror("allocation failure in vector()");
        return v-nl;
}

/* DEFINE MATRIX */
float **matrix(int nrl, int nrh, int ncl, int nch)
{
        int i;
        float **m;

        m=(float **) malloc((nrh-nrl+1)*sizeof(float*));
        if (!m) nrerror("allocation failure 1 in matrix()");
        m -= nrl;

        for(i=nrl;i<=nrh;i++) {
                m[i]=(float *) malloc((unsigned) (nch-ncl+1)*sizeof(float));
                if (!m[i]) nrerror("allocation failure 2 in matrix()");
                m[i] -= ncl;
        }
        return m;
}

/* FREE VECTOR */
void free_vector(v,nl,nh)
float *v;
int nl,nh;
{
        free((char*) (v+nl));
}

/* FREE MATRIX */
void free_matrix(m,nrl,nrh,ncl,nch)
float **m;
int nrl,nrh,ncl,nch;
{
        int i;

        for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));
        free((char*) (m+nrl));
}

/* ERROR MESSAGE */
void nrerror(error_text)
char error_text[];
```

```
{
        void exit();

        fprintf(stderr,"Numerical Recipes run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
}

/* GET STRING FROM THE TEXT MODE */
char *GetaString(char *String, char *Prompt)
{
  char tbuf[80];
  char format[127], *ptr;
  int i;

  ptr = format;
  strcpy(format, "%[^");
  ptr += strlen(format);

  for (i = 1; i <= 31; i++) {

    *ptr = (char) i;
    ptr++;
  } /*  for  */
  strcpy(ptr, "]");

  tbuf[0] = 0;

  if (Prompt) printf("%s", Prompt);
  scanf(format, tbuf);
  i=getchar();

  i = strlen(tbuf);
  if (!i) return (0);

  String = (char*) malloc ((i + 1) * sizeof(char));
  strcpy(String, tbuf);

  return (String);
}

/* FUNCTIONS TO APPLY POINTS MATCHING AND SVD LEAST SQUARE
MINIMIZATION ALGORITHM */
```

```
void ApplyProcrustesTransform(npoints, PointList2, Center, Matrix, Translation,
    Scale, outpoints)
int npoints;
XYZPOINT *PointList2, *outpoints;
XYZPOINT Center, Translation;
float Scale;
NRMATRIX Matrix;
{
  int i, j;
  NRMATRIX a, p1, t2, r2, p2;
  NRVECTOR cent, trans;

  cent = vector(1,3);
  trans = vector(1,3);

  for (i = 1; i <= 3; i++) {
    cent[i] = Center[i-1];
    trans[i] = Translation[i-1];
  } /* for (i = 1; i <= 3; i++) */

  a = LoadNRmatrix(Matrix, 3, 3);
  p2 = LoadNRmatrix(PointList2, npoints, 3);

  NRrowShift(p2, trans, 1, npoints, 1, 3, 0);  /* add "Translation" */
  NRrowShift(p2, cent, 1, npoints, 1, 3, 1);   /* subtract "Center" */
  NRscale(p2, Scale, 1, npoints, 1, 3);

  t2 = NRmatrixTranspose(p2, 1, npoints, 1, 3);
  r2 = NRmatrixProduct(a, t2, 3, 3, npoints);

  free_matrix(p2, 1, npoints, 1, 3);

  p2 = NRmatrixTranspose(r2, 1, 3, 1, npoints);

  NRrowShift(p2, cent, 1, npoints, 1, 3, 0);  /* add "Center" back */

  UnloadNRmatrix(p2, 1, npoints, 1, 3, outpoints);

  free_matrix(p2, 1, npoints, 1, 3);
  free_matrix(a, 1, 3, 1, 3);
  free_matrix(t2, 1, 3, 1, npoints);
  free_matrix(r2, 1, 3, 1, npoints);

  free_vector(cent, 1, 3);
  free_vector(trans, 1, 3);
}

/* PROGRAM USING SVD ALGORITHM FROM GOLUB et al. */
void ProcrustesMatch(npoints, PointList1, PointList2, Center, Matrix,
     Translation, Scale, MeanError, MeanSquareError, RMSError, doscale,
     ifprint)
int npoints;
XYZPOINT *PointList1, *PointList2;
XYZPOINT Center;
/*float Matrix[3][3], *Scale;*/
NRMATRIX Matrix;
float *Scale;
XYZPOINT Translation;
int doscale, ifprint;
float *MeanError, *MeanSquareError, *RMSError;

/*  Procrustes:  finds the rigid body transformation which matches a 3D
    set of points to a homologous set, minimizing the mean squared mismatch
    between the point pairs.  The transformation found fits PointList1
    to PointList2.  To apply it, the caller must do the following:

      1)  Add "Translation" to the PointList2 space coordinates
      2)  Rotate by "Matrix" about "Center", i.e. subtract Center, rotate
          by Matrix, add Center back again.

    This function optionally prints out an analysis of the point-by-point
    misfits after fitting.

    Algorithm is as given by Schonemann, Psychometrika vol 35, p. 245 (1970).
    (see eq. 2.11).

    Needs to link with svdcmp and nrutil from the Numerical Recipes in
    C package.      */

{
  int i,j,k,l,m;
  XYZPOINT Centroid1, Centroid2;
  NRMATRIX  a, b, u, v, r, p1, p2, t1, t2, r1, delta;
  NRMATRIX norm1, norm2;
  NRMATRIX  part1, part2, vt, ut;
  NRVECTOR w, distance;
  float mean, ms, rms, scale;
```

```
 if (ifprint) {

 printf("Procrustes: %d\n", npoints);

 for (i = 0; i < npoints; i++) {

   for (j = 0; j < 3; j++) printf("%6.2f ", PointList1[i][j]);
   for (j = 0; j < 3; j++) printf("%6.2f ", PointList2[i][j]);
   printf("\n");
 } /* for (i = 0; i < npoints; i++)  */
 } /*   if (ifprint)  */

/*  SVD ALGORITHM BEGINS */

 a = matrix(1, 3, 1, 3);
 w = vector(1, 3);
 v = matrix(1, 3, 1, 3);

 p1 = LoadNRmatrix(PointList1, npoints, 3);
 p2 = LoadNRmatrix(PointList2, npoints, 3);

   for (j = 0; j < 3; j++) {
     Centroid1[j] = 0.0;
     Centroid2[j] = 0.0;
   }

  for (i = 0; i < npoints; i++)
   for (j = 0; j < 3; j++) {
     Centroid1[j] += PointList1[i][j];
     Centroid2[j] += PointList2[i][j];
   }

   for (j = 0; j < 3; j++) {
     Centroid1[j] /= npoints;
     Centroid2[j] /= npoints;
   }

 for (i = 1; i <= npoints; i++) {

   for (j = 1; j <= 3; j++) {
     p1[i][j] -= Centroid1[j-1];
     p2[i][j] -= Centroid2[j-1];
   } /*   for (j = 1; j <= 3; j++)  */
} /*  for (i = 1; i <= npoints; i++)  */

for (i = 1; i <= 3; i++)
  for (j = 1; j <= 3; j++)  a[i][j] = 0.0;

    for (j = 1; j <= npoints; j++) {
      for (k = 1; k <= 3; k++)
        for (l = 1; l <= 3; l++)
          a[k][l] += p2[j][k] * p1[j][l];
    }

svdcmp(a, 3, 3, w, v);

u = a;

ut = NRmatrixTranspose(u, 1, 3, 1, 3);
vt = NRmatrixTranspose(v, 1, 3, 1, 3);

r = NRmatrixProduct(v, ut, 3, 3, 3);

if (ifprint)
    PrintNRmatrix("This is the rotation matrix: \n", r, 1, 3, 1, 3, 1);

t1 = NRmatrixTranspose(p2, 1, npoints, 1, 3);
norm1 = NRmatrixProduct(p2, t1, npoints, 3, npoints);  /* B x B' */

t2 = NRmatrixTranspose(p1, 1, npoints, 1, 3);

r1 = NRmatrixProduct(r, t1, 3, 3, npoints);
norm2 = NRmatrixProduct(p1, r1,  npoints, 3, npoints); /* A x RB' */

*Scale = scale = NRtrace(norm2, 1, npoints) / NRtrace(norm1, 1, npoints);
  /* scale according to Schonemann & Carroll's recipe */
  /* eq. 2.12, p. 247. */

printf("scale is %.2f\n", scale);

for (i = 1; i <= npoints; i++) {

  for (j = 1; j <= 3; j++) {
    if (doscale) r1[j][i] *= scale;
    r1[j][i] += Centroid1[j-1];
    t2[j][i] += Centroid1[j-1];
  } /*    for (j = 1; j <= 3; j++)  */
  } /*  for (i = 1; i <= npoints; i++)  */

  if (ifprint) {

  printf("Centroid 1 is ");for (i = 0; i < 3; i++) printf("%.2f ", Centroid1[i]);
printf("\n");
  printf("Centroid 2 is ");for (i = 0; i < 3; i++) printf("%.2f ", Centroid2[i]);
printf("\n");
  } /*  if (ifprint)  */

  mean = ms = 0.0;

  delta = NRmatrixSubtract(t2, r1, 1, 3, 1, npoints);
  distance = NRrowColNorm(delta, 1, 3, 1, npoints, 0);

  for (i = 1; i <= npoints; i++) {

  if (ifprint)
    printf("(%6.2f, %6.2f, %6.2f)     %.2f\n",
       delta[1][i], delta[2][i], delta[3][i], distance[i]);

    mean += distance[i];
    ms += distance[i] * distance[i];

  } /* for (i = 0; i <= npoints; i++)  */

  mean /= npoints;
  ms /= npoints;
  rms = (float) sqrt((double)ms);

  *MeanError = mean;    /* return residual values to caller */
  *MeanSquareError = ms;
  *RMSError = rms;

  if (ifprint)
  printf("mean distance = %.2f; mean square distance = %.2f; rms distance = %.2f\n",
mean, ms, rms);

  UnloadNRmatrix(r, 1, 3, 1, 3, Matrix);
  for (i = 0; i < 3; i++) {

    Center[i] = Centroid1[i];
    Translation[i] = Centroid1[i] - Centroid2[i];
  } /* for (i = 1; i <= 3; i++)  */

  free_matrix(p1, 1, npoints, 1, 3);
  free_matrix(p2, 1, npoints, 1, 3);

  free_matrix(t1, 1, 3, 1, npoints);
  free_matrix(t2, 1, 3, 1, npoints);

  free_matrix(r, 1, 3, 1, 3);
  free_matrix(r1, 1, 3, 1, npoints);

  free_matrix(norm1, 1, npoints, 1, npoints);
  free_matrix(norm2, 1, npoints, 1, npoints);

  free_matrix(a, 1, 3, 1, 3);
  free_matrix(ut, 1, 3, 1, 3);
  free_matrix(vt, 1, 3, 1, 3);
  free_matrix(v, 1, 3, 1, 3);

  free_matrix(delta, 1, 3, 1, npoints);
  free_vector(distance, 1, npoints);

  return;
}

/* MAIN LOOP TO RUN THE POINT MATCHING ALGORITHM */
static int GetInt(fd, IntVal)
int *IntVal;
FILE *fd;

{
  *IntVal = 0;
  fscanf(fd, "%d", IntVal);

  return (*IntVal );
}

/*  prompts for file name, reads in  two lists of 3D points into two NRUTIL vectors
(which it also allocates).  returns a pointer to  the file name or NULL if none.  */

static int ReadPointFile(infile, list1, list2)
 XYZPOINT **list1, **list2;
 char *infile;
{
   char *filename;  /* buffer for the file name  */
   int i;
   int npoints;
   char reply[128];

   if (infile) return(ReadPointData(infile, list1, list2));

   i = 0;
   while (i == 0) {

     if ((filename = GetaString(reply, "Name of points file: ")) == NULL) return 0;

     i = ReadPointData(filename, list1, list2);
     free(filename);

   } /* while (i == 0)    */

  return(i);
}   /* end ReadPointFile  */

static int ReadPointData( filename,list1, list2)
 XYZPOINT **list1, **list2;
 char *filename; /* the file to be read from */
{
  int i, j, n, npts;
  FILE *fd;
  float *fptr;
  char inbuf[132], *ptr, *end;
  XYZPOINT *buf1;

  if (  ( fd = fopen(filename, "r") ) != NULL)  {  /* found the file - read it  */

    if (  GetInt(fd, &npts) > 0 )  {  /*  if any points in this file  */

      printf("going to read %d points \n", npts);
      buf1 = (XYZPOINT *) malloc(2 * npts * sizeof(XYZPOINT));
      /* space for coordinates*/

      n = 0;

      fgets(inbuf, sizeof(inbuf) - 1, fd);
      fptr = (float *) buf1;

      while (n < 6 * npts) {
        /* read in funny way to accomodate both blank and comma delimiters */
        fgets(inbuf, sizeof(inbuf) - 1, fd);
        end = inbuf + strlen(inbuf);
         *(end + 1) = ' ';
        ptr = inbuf;
        while (n < 6 * npts && ptr < end)  {
          sscanf(ptr, "%f", fptr);
   /*     printf("%f\n", *fptr);        */
          fptr++;
         n++;
          ptr = strpbrk(ptr + 1, ", \n") + 1;

        } /* while (i = sscanf(inbuf, "%f", fptr) )  */

       *list1 = (XYZPOINT *) malloc (npts * sizeof (XYZPOINT));
       *list2 = (XYZPOINT *) malloc (npts * sizeof (XYZPOINT));

       for (i = 0; i < npts; i++) {  /* calculate centroids, load lists  */

         for (j = 0; j < 3; j++) {

           (*list1)[i][j] = buf1[2*i][j];
           (*list2)[i][j] = buf1[2*i+1][j];

         }
       }   /*  loop over points - read in and calculate centroid  */

     }  /* if any points in this hat  */

    fclose(fd);
    free(buf1);

      return ( npts );

   }  /* if opened the file ok  */

   else
     printf("could not open file %s\n", filename);
     return(0);

   }   /* end ReadPointData  */
}

/* PROGRAM MAIN FUNCTION HERE */
```

```
main(int argc, char **argv)
{
 XYZPOINT *PointList1, *PointList2;
 XYZPOINT *outpoints;
 int npoints;
 float amat[3][3], scale;
 float xx, yy, zz;
 XYZPOINT origin, shift;
 int i, j, doscale;
 int mni=0;
 float mean, ms, rms;
 char *answer;
 char *outfile = NULL;
 char *infile = NULL;
 FILE *fd;

 clrscr();

  for (i = 1; i < argc; i++) {

   if ( (i == 1) && (argv[i][0] == '?')) {

     printf("Usage: pros [inputfile] [-scale] [-mni] [-out outfile]\n");
     exit(0);

   } /*  if (!strcmp(argv[i], "-out")  */

   else if ( (i == 1) && (argv[i][0] != '-')) {

     infile = (char *)malloc((strlen(argv[i]) + 1) * sizeof(char));
     strcpy(infile, argv[i]);

   } /*  if (!strcmp(argv[i], "-out")  */
   else if (!strcmp(argv[i], "-MNI") || !strcmp(argv[i], "-mni")) mni = 1;
   else if (!strcmp(argv[i], "-scale")) doscale = 1;
   else if ( (i + 1 < argc) && !strcmp(argv[i], "-out")) {

     outfile = (char *)malloc((strlen(argv[i+1]) + 1) * sizeof(char));
     strcpy(outfile, argv[i+1]);

   } /*  if (!strcmp(argv[i], "-out")  */

  } /* for */
 if (! (npoints = ReadPointFile(infile, &PointList1, &PointList2 ))) return(1);

 printf("readpointfile returned %d points\n", npoints);

  if (mni) for (i = 0; i < npoints; i++){

     PointList1[i][1] *= -1;
     PointList2[i][1] *= -1;

  } /*  if (mni)  */


 ProcrustesMatch(npoints, PointList1, PointList2, origin, amat, shift,
  &scale, &mean, &ms, &rms, doscale, 0);

 printf("Procrustes returned matrix:\n");
 for (i = 0; i < 3; i++)  {

  for (j = 0; j < 3; j++) printf("%6.2f ", amat[i][j]);
  printf("\n");

 }

 solve_euler(amat,&xx,&yy,&zz);

 printf("Three euler angles are (Z-Y-X axis order): %f %f %f\n", xx, yy, zz);

 printf("Procrustes returned origin:\n");

  for (j = 0; j < 3; j++) printf("%6.2f ", origin[j]);
  printf("\n");

 printf("Procrustes returned shift:\n");

  for (j = 0; j < 3; j++) printf("%6.2f ", shift[j]);
  printf("\n");

 printf("Procrustes returned scale %6.3f\n", scale);

 printf("Procrustes returned mean, ms, rms errors %6.2f, %6.2f, %6.2f\n",
  mean, ms, rms);

outpoints = (XYZPOINT *) malloc (npoints * sizeof(XYZPOINT));
```

```
 if (!doscale) scale = 1.0;

  if (outfile) {
    fd = fopen(outfile,"w");
    fprintf(fd, "origin: %.3f %.3f %.3f\n", origin[0], origin[1], origin[2]);
    fprintf(fd, "shift: %.3f %.3f %.3f\n", shift[0], shift[1], shift[2]);
    fprintf(fd, "scale: %.3f\n", scale);
    fprintf(fd, "matrix:\n");

    for (i = 0; i < 3; i++)  {

      for (j = 0; j < 3; j++) fprintf(fd,"%f ", amat[i][j]);
      fprintf(fd, "\n");

    };

     fprintf(fd, "Three euler angles are (Z-Y-X axis order): %.3f %.3f %.3f\n", xx, yy,
zz);

  } /*   if (outfile)  */

 ApplyProcrustesTransform(npoints, PointList2, origin, amat, shift,
    scale, outpoints);

    if (mni) for (i = 0; i < npoints; i++){

       PointList1[i][1] *= -1;
       PointList2[i][1] *= -1;
       outpoints[i][1] *= -1;

    } /*     if (mni)  */

 for (i = 0; i < npoints; i++) {

    printf("(%6.2f, %6.2f, %6.2f)  (%6.2f, %6.2f, %6.2f) (%6.2f, %6.2f, %6.2f)\n",
      PointList1[i][0], PointList1[i][1], PointList1[i][2],
      outpoints[i][0], outpoints[i][1], outpoints[i][2],
      PointList1[i][0] - outpoints[i][0], PointList1[i][1] - outpoints[i][1],
       PointList1[i][2] - outpoints[i][2]);

    if (outfile)
    fprintf(fd,
      "(%6.2f, %6.2f, %6.2f)  (%6.2f, %6.2f, %6.2f) (%6.2f, %6.2f, %6.2f)\n",
      PointList1[i][0], PointList1[i][1], PointList1[i][2],
    outpoints[i][0], outpoints[i][1], outpoints[i][2],
    PointList1[i][0] - outpoints[i][0], PointList1[i][1] - outpoints[i][1],
     PointList1[i][2] - outpoints[i][2]);

  } /*  for (i = 0; i < npoints; i++)  */

  if (outfile)
    fprintf(fd, "mean, ms, rms errors: %6.2f, %6.2f, %6.2f\n",
   mean, ms, rms);

  if (outfile) fclose(fd);

  return(0);

}

/*******************************************************************
*******/
/*  SVD ALGORITHM FROM NUMERICAL RECIPE */
/*******************************************************************
*******/

void svdcmp(float **a, int m, int n, float *w, float **v)
{
        int flag,i,its,j,jj,k,l,nm;
        float c,f,h,s,x,y,z;
        float anorm=0.0,g=0.0,scale=0.0;
        float *rv1,*vector();
        void nrerror(),free_vector();

        if (m < n) nrerror("SVDCMP: You must augment A with extra zero rows");
        rv1=vector(1,n);
        for (i=1;i<=n;i++) {
                l=i+1;
                rv1[i]=scale*g;
                g=s=scale=0.0;
                if (i <= m) {
                        for (k=i;k<=m;k++) scale += fabs(a[k][i]);
                        if (scale) {
                                for (k=i;k<=m;k++) {
                                        a[k][i] /= scale;
                                        s += a[k][i]*a[k][i];
                                }
                                f=a[i][i];
```

```
                    g = -SIGN(sqrt(s),f);
                    h=f*g-s;
                    a[i][i]=f-g;
                    if (i != n) {
                            for (j=l;j<=n;j++) {
                                    for   (s=0.0,k=i;k<=m;k++)   s   +=
a[k][i]*a[k][j];

                                    f=s/h;
                                    for (k=i;k<=m;k++) a[k][j] += f*a[k][i];
                            }
                    }
                    for (k=i;k<=m;k++) a[k][i] *= scale;
            }
        }
        w[i]=scale*g;
        g=s=scale=0.0;
        if (i <= m && i != n) {
                for (k=l;k<=n;k++) scale += fabs(a[i][k]);
                if (scale) {
                        for (k=l;k<=n;k++) {
                                a[i][k] /= scale;
                                s += a[i][k]*a[i][k];
                        }
                        f=a[i][l];
                        g = -SIGN(sqrt(s),f);
                        h=f*g-s;
                        a[i][l]=f-g;
                        for (k=l;k<=n;k++) rv1[k]=a[i][k]/h;
                        if (i != m) {
                                for (j=l;j<=m;j++) {
                                        for   (s=0.0,k=l;k<=n;k++)   s   +=
a[j][k]*a[i][k];

                                        for (k=l;k<=n;k++) a[j][k] += s*rv1[k];
                                }
                        }
                        for (k=l;k<=n;k++) a[i][k] *= scale;
                }
        }
        anorm=MAX(anorm,(fabs(w[i])+fabs(rv1[i])));
    }
    for (i=n;i>=1;i--) {
        if (i < n) {
                if (g) {
                        for (j=l;j<=n;j++)
                    v[j][i]=(a[i][j]/a[i][l])/g;
                for (j=l;j<=n;j++) {
                    for(s=0.0,k=l;k<=n;k++)s+=a[i][k]*v[k][j];
                    for (k=l;k<=n;k++) v[k][j] += s*v[k][i];
                }
            }
            for (j=l;j<=n;j++) v[i][j]=v[j][i]=0.0;
        }
        v[i][i]=1.0;
        g=rv1[i];
        l=i;
    }
    for (i=n;i>=1;i--) {
        l=i+1;
        g=w[i];
        if (i < n)
            for (j=l;j<=n;j++) a[i][j]=0.0;
        if (g) {
            g=1.0/g;
            if (i != n) {
                for (j=l;j<=n;j++) {
                    for (s=0.0,k=l;k<=m;k++) s += a[k][i]*a[k][j];
                    f=(s/a[i][i])*g;
                    for (k=i;k<=m;k++) a[k][j] += f*a[k][i];
                }
            }
            for (j=i;j<=m;j++) a[j][i] *= g;
        } else {
            for (j=i;j<=m;j++) a[j][i]=0.0;
        }
        ++a[i][i];
    }
    for (k=n;k>=1;k--) {
        for (its=1;its<=30;its++) {
            flag=1;
            for (l=k;l>=1;l--) {
                nm=l-1;
                if (fabs(rv1[l])+anorm == anorm) {
                    flag=0;
                    break;
                }
                if (fabs(w[nm])+anorm == anorm) break;
            }
            if (flag) {
                c=0.0;
                s=1.0;
                for (i=l;i<=k;i++) {
                        f=s*rv1[i];
                        if (fabs(f)+anorm != anorm) {
                                g=w[i];
                                h=PYTHAG(f,g);
                                w[i]=h;
                                h=1.0/h;
                                c=g*h;
                                s=(-f*h);
                                for (j=1;j<=m;j++) {
                                        y=a[j][nm];
                                        z=a[j][i];
                                        a[j][nm]=y*c+z*s;
                                        a[j][i]=z*c-y*s;
                                }
                        }
                }
        }
        z=w[k];
        if (l == k) {
                if (z < 0.0) {
                        w[k] = -z;
                        for (j=1;j<=n;j++) v[j][k]=(-v[j][k]);
                }
                break;
        }
        if (its == 30) nrerror("No convergence in 30 SVDCMP
iterations");
        x=w[l];
        nm=k-1;
        y=w[nm];
        g=rv1[nm];
        h=rv1[k];
        f=((y-z)*(y+z)+(g-h)*(g+h))/(2.0*h*y);
        g=PYTHAG(f,1.0);
        f=((x-z)*(x+z)+h*((y/(f+SIGN(g,f)))-h))/x;
        c=s=1.0;
        for (j=l;j<=nm;j++) {
                i=j+1;
                g=rv1[i];
                y=w[i];
                h=s*g;
```

```
                    g=c*g;
                    z=PYTHAG(f,h);
                    rv1[j]=z;
                    c=f/z;
                    s=h/z;
                    f=x*c+g*s;
                    g=g*c-x*s;
                    h=y*s;
                    y=y*c;
                    for (jj=1;jj<=n;jj++) {
                        x=v[jj][j];
                        z=v[jj][i];
                        v[jj][j]=x*c+z*s;
                        v[jj][i]=z*c-x*s;
                    }
                    z=PYTHAG(f,h);
                    w[j]=z;
                    if (z) {
                        z=1.0/z;
                        c=f*z;
                        s=h*z;
                    }
                    f=(c*g)+(s*y);
                    x=(c*y)-(s*g);
                    for (jj=1;jj<=m;jj++) {
                        y=a[jj][j];
                        z=a[jj][i];
                        a[jj][j]=y*c+z*s;
                        a[jj][i]=z*c-y*s;
                    }
                }
                rv1[l]=0.0;
                rv1[k]=f;
                w[k]=x;
            }
        }
        free_vector(rv1,1,n);
}

#undef SIGN
#undef MAX
#undef PYTHAG

void svbksb(u,w,v,m,n,b,x)
```

```
float **u,w[],**v,b[],x[];
int m,n;
{
        int jj,j,i;
        float s,*tmp,*vector();
        void free_vector();

        tmp = vector(1,n);
        for (j=1;j<=n;j++) {
                s=0.0;
                if (w[j]) {
                        for (i=1;i<=m;i++) s += u[i][j]*b[i];
                        s /= w[j];
                }
                tmp[j] = s;
        }
        for (j=1;j<=n;j++) {
                s=0.0;
                for (jj=1;jj<=n;jj++) s += v[j][jj]*tmp[jj];
                x[j] = s;
        }
        free_vector(tmp,1,n);
}

/***********************************************************************
*******/
/* THIS SUBROUTINE IS TO SOLVE THREE EULER ANGLES */
/***********************************************************************
*******/
void solve_euler(float amat[3][3], float *x, float *y, float *z)
{

  int i,j;
  float amat1[3][3], ty;

//  ty = -(float)asin((double)amat[0][2]); Not conventional setting
    ty = -(float)asin((double)amat[2][0]);

  if  (fabs( fabs((double)ty) - PI/2.0) < 0.001)
            /*(ty == PI/2.0 || ty == -PI/2.0)*/  {

/*   eliminate degeneracy in x and z when y is + or - 90 degrees */

    rotY(A_SIGN(0.01,-ty), amat1);
    Mult3x3(amat1, amat, amat);
    for (i = 0; i < 3; i++) {
      for (j = 0; j < 3; printf("%f ",amat[i][j]), j++); printf("\n");
    }

  } /* if (y == PI/2.0 || y == PI/2.0)   */

// The followings are not the conventional settings
//  *x = (float) atan2( (double) amat[1][2], (double) amat[2][2]) / RADIAN;
//  *y = ty / RADIAN;
//  *z = (float) atan2( (double) amat[0][1], (double) amat[0][0]) / RADIAN;

  *x = (float) atan2( (double) amat[1][0], (double) amat[0][0]) / RADIAN;
  *y = ty / RADIAN;
  *z = (float) atan2( (double) amat[2][1], (double) amat[2][2]) / RADIAN;

  if ( fabs(fabs((double) (*x)) -180.0) < 1.0e-05 &&
    fabs(fabs((double) (*z)) -180.0) < 1.0e-05) {

/* if both x and z are 180 degrees, it is the same as if they were zero, and
  y were 180 minus its present value.  that is a lot neater, so force it.  */

    *x = 0.0;
    *z = 0.0;
    *y = A_SIGN(180.0, (*y)) - (*y);

  } /*     fabs(fabs(double) *z) -180.0) < 1.0e-05)  */
}

void rotY(float thet, float amat[3][3])
{

float ct, st;
int i;

ct = (float) cos((double)thet * RADIAN);
st = (float) sin((double)thet * RADIAN);

amat[0][0] = ct;
amat[0][2] = -st;

amat[2][0] = st;
amat[2][2] = ct;
```

```
for(i = 0; i < 3; amat[1][i] = amat[i][1] = 0, i++);

amat[1][1] = 1.0;

return;
}


void Mult3x3(float amat[3][3], float bmat[3][3], float cmat[3][3])
{
  float scratch[3][3];
  int i, j, k;

  for (i = 0; i < 3; i++)  {

    for (j = 0; j < 3; j++)  {

      scratch[i][j] = 0.0;

      for (k = 0; k < 3; scratch[i][j] += amat[i][k] * bmat[k][j], k++);
    }
  }

  for (i = 0; i < 3; i++) {

    for (j = 0; j < 3; cmat[i][j] = scratch[i][j], j++);
  }

return;

}
```

```
/***************************************************************************/
/*                  HOOKE AND JEEVE OPTIMIZATION ALGORITHM              */
/***************************************************************************/
/* This program is the Hooke and Jeeves optimization algorithm
   for calculation of the six degree-of-freedom fitting
   parameters to realign the patient's fractionated treatment
   position.  This program requires the 3-D input data set which contains
   number of the total 3-D points (1st line) and the X-Y-Z coordinates
   of the two sets of points separated by space.  After calculation the
   parameters are returned either as a print out format or file format.
   The output file can be viewed from any kind of text editor.
   edited by Jack Yang at the Shands Cancer Center, Jan 15, 1994.
*/

/* Include all the Header Files */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

/* Dimensions of the appratus ( the true dimensions may vary ) */
/* The initial values are assumed zeros because of the Euler angle definition */
#define T1 0.0
#define T2 0.0
#define T3 0.0
#define T4 0.0
#define T5 0.0
#define T6 0.0
#define T7 0.0
#define T8 0.0
#define T9 0.0

#define radtodeg 57.29578       /* Transfer from radian to degree */
#define degtorad 0.0174532      /* Transfer from degree to radian */
#define SQ(x) ((x)*(x))
#define N 500                   /* Define the maximum number of points */

/* Function prototypes */
 FILE *gfopen(char *fn, char *mode);
 void TRANSLATION(float data1[N][4], float data2[N][4]);
 void NORMAL(float A[N][3]);
 void TRANSFORM(float T[3], float old[N][4], float new[N][4]);
 void obj_fun(float T[3], float *f);
 void HK_JV(float t[3]);
 void prn_matrix(float angle[3], FILE *outfp);

/* Definition of global variables */
/*------------------------------------------------------------------*/
/* points_1r : 1st day, BRW or any specified coordinate */
/* points_1f : 1st day, fixed coordinate */
/* points_2r : 2nd day, BRW or any specified coordinate */
/* points_2f : 2nd day, fixed coordinate */
/* vector_1  : 1st day vectors to each point (in the fixed system) */
/* vector_2  : 2nd day vectors to each point (in the fixed system) */
/* offset    : parameter for calculating mean error, mean squared error and
              root mean squared error */
/* shift     : translation vector for patient realignment*/
/* center1   : The centroid of 1st point sets before transformation*/
/* center2   : The centroid of 2nd point sets after transformation*/
/*------------------------------------------------------------------*/
float points_1r[N][4], points_1f[N][4], points_2r[N][4], points_2f[N][4];
float vector_1[N][3], vector_2[N][3], theta_1[3], theta_2[3];
float offset[N][3], shift[3], shift1[3];
float center1[3], center2[3], center_temp[3];
int numpoints = 0;          /* Number of point counts */

/* MAIN PROGRAM STARTS HERE */
void main(int argc, char *argv[])
{
 int i, j, k;
 float d1[3];
 float norm[N], mean, mean_square, rms;
 FILE *fp, *outfp;

 clrscr();

 if(argc != 3)
  {
   fprintf(stdout, "\n** You forgot to specify file name **\n");
   fprintf(stderr, "\n%s%s%s%s\n\n%s\n\n",
           "Usage:  ", argv[0], " input_file ", " output_file ",
           "(Ex: hooke [input file] [output file])" );
   exit(1);
  }
 fp = gfopen(argv[1], "r+");
 outfp = gfopen(argv[2], "w");

 fscanf(fp, "%d", &numpoints);
```

```
        printf("Number of points = %d\n", numpoints);

    printf("\n ***** data sets ***** -- File: %s --\n",*(argv+1));
    for(i = 0; i < numpoints; i++) {
        points_1r[i][3] = 1.0;
        points_2r[i][3] = 1.0;
        fscanf(fp, "%f %f %f %f %f %f", &points_1r[i][0],
            &points_1r[i][1], &points_1r[i][2], &points_2r[i][0],
            &points_2r[i][1], &points_2r[i][2]);
    }

    printf("\n***** 1st day data *****        ***** 2nd day data *****\n");
    for(i = 0; i < numpoints; i++)
     printf("(%8.4f %8.4f %8.4f)  (%8.4f %8.4f %8.4f)\n", points_1r[i][0],
         points_1r[i][1], points_1r[i][2], points_2r[i][0], points_2r[i][1],
         points_2r[i][2]);

/* Now perform the translation of the two different data sets ---
   First, move the two point sets to the (0,0,0) of the ring system
   or any specified coordinate system.
   Second, return the calculated translation vector.
*/

    TRANSLATION(points_1r, points_2r);

    for(i = 0; i < numpoints; i++) {
      for(j = 0; j < 3; j++) {
        points_1r[i][j] -= center1[j];
        points_2r[i][j] -= center2[j];
      }
     }

/* Print out the linear settings of the setups.
   Assume that the linear readings are zero  at this point */
    printf("\nLinear readings (longitudinal, transverse, and vertical):\n");
    for(i = 0; i < 3; i++) dl[i] = 0.0;
    printf("%8.4f %8.4f %8.4f\n",dl[0],dl[1],dl[2]);

    /* Assume that the 1st day angles are all set to zero value */
    for(i = 0; i < 3 ; i++) theta_1[i] = 0.0;

    TRANSFORM(theta_1,points_1r,points_1f);

    /* Calculate the vectors for 1st day which related to the absolute origin
    (Assume that all vectors refer to the origin point) */
  for(i = 0; i < numpoints; i++) {
   vector_1[i][0] = points_1f[i][0] ;
   vector_1[i][1] = points_1f[i][1] ;
   vector_1[i][2] = points_1f[i][2] ;
  }

  NORMAL(vector_1); /* Normalize the above vectors to unitary vectors */

  /* Input the initial value for optimization and perform optimization,
     the outputs are calculated */
  HK_JV(theta_1);

 /* Transform 2nd data sets to the fixed system w/ the calculated theta_2[3] */
  TRANSFORM(theta_2,points_2r,points_2f);

  for(i = 0; i < numpoints; i++) {
   for(j = 0; j < 3; j++) {
          points_1f[i][j] += center1[j];
          points_2r[i][j] += center2[j];
   }
  }

 /* Transform 2nd data sets to the fixed system w/ the calculated theta_2[3] */
  TRANSFORM(theta_2,points_2r,points_2f);

  for(j = 0; j < 3; j++) {
        center_temp[j] = 0.0;
  }

  TRANSFORM1(theta_2, center2, center_temp);

  for(j = 0; j < 3; j++) {
        shift1[j] = center1[j] - center_temp[j];
 /*     printf("****center1[%d] = %4.2f,
               center_temp[%d]=%4.2f\n",j,center1[j],j,center_temp[j]);
        printf("shift1[%d] = %f\n", j, shift1[j]); */
  }

/* Re-calculate the translation vector of the
   least-squared minimization point sets */

/* Add the translation vector of point sets #2 for statistical analysis */
 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 3; j++) {
        points_2f[i][j] += shift1[j];
  }
 }

/* Print out the calculated parameters to the monitor */
/*-------------------------------------------------------------------------------*/
 printf("\n                    ***** Displacements for points *****");
 printf("\n(Longitudinal, Transverse, Vertical)");
 for(i = 0; i < numpoints; i++) {
  offset[i][0] = d1[0] - (points_2f[i][0] - points_1f[i][0]); /* in longitudinal direction */
  offset[i][1] = d1[1] - (points_2f[i][1] - points_1f[i][1]); /* in transverse direction */
  offset[i][2] = d1[2] - (points_2f[i][2] - points_1f[i][2]); /* in vertical direction */

 }

/* Statistical analysis */
/* Calculate the mean, mean square, and rms errors of the data points */
  mean = mean_square = 0.0;
  for(i = 0; i < numpoints; i++) {
        norm[i] = 0.0;
    for(j = 0; j < 3; j++) {
        norm[i] += SQ(offset[i][j]);
       norm[i] = sqrt(norm[i]);
    }
  mean += norm[i];
  mean_square += SQ(norm[i]);
  }

  mean /= numpoints;
  mean_square /= numpoints;
  rms = sqrt(mean_square);

 /* Print out the calculated parameters to the monitor */
 printf("\n\n                    ***** RESULT *****");
 printf("\nDisplacement in longitudinal direction (X) : %8.3lf",shift1[0]);
 printf("\nDisplacement in transverse direction   (Y) : %8.3lf",shift1[1]);
 printf("\nDisplacement in vertivcal direction   (Z) : %8.3lf",shift1[2]);
 printf("\nRotation of joint 4 (Euler definition: Z-Yaw)     : %8.3lf
deg",theta_2[0]*radtodeg);
 printf("\nRotation of joint 5 (Euler definition: Y-Pitch)   : %8.3lf
deg",theta_2[1]*radtodeg);
 printf("\nRotation of joint 6 (Euler definition: X-Roll)    : %8.3lf
deg\n",theta_2[2]*radtodeg);
```

```
 printf("\nmean = %8.3lf mean_square = %8.3lf rms = %8.3lf\n", mean,
mean_square, rms);


/* Print the results to the output file */

 fprintf(outfp, "\n*** 1st day data ***          *** 2nd day data ***          ***
Displacements for points ***\n");
 for(i = 0; i < numpoints; i++)
  fprintf(outfp, "(%8.4f %8.4f %8.4f)(%8.4f %8.4f %8.4f)(%8.4f %8.4f %8.4f)\n",
points_1f[i][0],
        points_1f[i][1], points_1f[i][2], points_2f[i][0], points_2f[i][1],
        points_2f[i][2], offset[i][0], offset[i][1], offset[i][2]);

 fprintf(outfp, "\n\ncenter1[0][1][2] = %8.4f %8.4f %8.4f\n", center1[0], center1[1],
center1[2]);
 fprintf(outfp, "\n\ncenter2[0][1][2] = %8.4f %8.4f %8.4f\n", center2[0], center2[1],
center2[2]);

 fprintf(outfp, "\n\n***** RESULT *****\n");
  fprintf(outfp, "Displacement in longitudinal direction (X) : %8.3lf\n",shift1[0]);
  fprintf(outfp, "Displacement in transverse direction   (Y) : %8.3lf\n",shift1[1]);
  fprintf(outfp, "Displacement in vertivcal direction    (Z) : %8.3lf\n",shift1[2]);
  fprintf(outfp, "\nRotation of joint 4 (Euler definition: Z-Yaw)   : %8.3lf
deg",theta_2[0]*radtodeg);
  fprintf(outfp, "\nRotation of joint 5 (Euler definition: Y-Pitch) : %8.3lf
deg",theta_2[1]*radtodeg);
  fprintf(outfp, "\nRotation of joint 6 (Euler definition: X-Roll)  : %8.3lf
deg\n",theta_2[2]*radtodeg);
  fprintf(outfp, "\nmean = %8.3lf mean_square = %8.3lf rms = %8.3lf\n", mean,
mean_square, rms);

  prn_matrix(theta_2, outfp); /* Print out the matrix to verify the calculated results */

  fclose(fp);
  fclose(outfp);

 }

/******************/
/*  Subroutines */
/******************/
/* Function to normalize the vector */
void NORMAL(A)
```

```
float A[N][3];
{
 int i, j, k ;
 float s ;

 for(i = 0; i < numpoints; i++) {
  s = 0.0;
  for (j = 0; j < 3; j++)
    s += SQ(A[i][j]);
  for ( j = 0 ; j < 3 ; j++ )
   A[i][j] = A[i][j] / sqrt(s) ;
 }
}

/* This function is to transform the ring or BRW coordinates to the
   base coordinates with a combination of rotation and translation */
void TRANSFORM(T, old, new)
float T[3], old[N][4], new[N][4];
{
 int i, j;
 float c1, s1, c2, s2, c3, s3;
 float t_matrix[4][4];
 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 4; j++)
  new[i][j] = 0.0;
 }
 c1 = cos(T[0]);
 s1 = sin(T[0]);
 c2 = cos(T[1]);
 s2 = sin(T[1]);
 c3 = cos(T[2]);
 s3 = sin(T[2]);
 for(i = 0; i < 4 ; i++)
  {
   for( j = 0; j < 4; j++)
    t_matrix[i][j] = 0.0;
  }
 // The followings are the convention defined by the drawing
/* t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = -c1 * s2 * c3 + s1 * s3;
 t_matrix[0][2] = c1 * s2 * s3 + s1 * c3;
 t_matrix[0][3] = c1 * c2 * T4 + s1 * (T2 + T3);
 t_matrix[1][0] = s2;
 t_matrix[1][1] = c2 * c3;
t_matrix[1][2] = -c2 * s3;
t_matrix[1][3] = s2 * T4;
t_matrix[2][0] = -s1 * c2;
t_matrix[2][1] = s1 * s2 * c3 + c1 * s3;
t_matrix[2][2] = -s1 * s2 * s3 + c1 * c3;
t_matrix[2][3] = -s1 * c2 * T4 + c1 * (T2 + T3) + T1;
t_matrix[3][0] = 0.0;
t_matrix[3][1] = 0.0;
t_matrix[3][2] = 0.0;
t_matrix[3][3] = 1.0;                */

/* The followings are the Euler angles (Z-Y-X) with
   different rotational origins */
t_matrix[0][0] = c1 * c2;
t_matrix[0][1] = c1 * s2 * s3 - s1 * c3;
t_matrix[0][2] = c1 * s2 * c3 + s1 * s3;
t_matrix[0][3] = c1 * c2 * (T1+T4) + c1 * s2 * (T3+T6) + c1 * T7
                 - s1 * (T2+T5+T8);
t_matrix[1][0] = s1 * c2;
t_matrix[1][1] = s1 * s2 * s3 + c1 * c3;
t_matrix[1][2] = s1 * s2 * c3 - c1 * s3;
t_matrix[1][3] = s1 * c2 * (T1+T4) + s1 * s2 * (T3+T6) + s1 * T7
                 + c1 * (T2+T5+T8);
t_matrix[2][0] = -s2;
t_matrix[2][1] = c2 * s3;
t_matrix[2][2] = c2 * c3;
t_matrix[2][3] = -s2 * (T1+T4) + c2 * (T3+T6) + T9;
t_matrix[3][0] = 0.0;
t_matrix[3][1] = 0.0;
t_matrix[3][2] = 0.0;
t_matrix[3][3] = 1.0;

for(i = 0; i < numpoints; i++) {
new[i][0] = t_matrix[0][0] * old[i][0] + t_matrix[0][1] * old[i][1] +
           t_matrix[0][2] * old[i][2] + t_matrix[0][3] ;
new[i][1] = t_matrix[1][0] * old[i][0] + t_matrix[1][1] * old[i][1] +
           t_matrix[1][2] * old[i][2] + t_matrix[1][3] ;
new[i][2] = t_matrix[2][0] * old[i][0] + t_matrix[2][1] * old[i][1] +
           t_matrix[2][2] * old[i][2] + t_matrix[2][3] ;
new[i][3] = 1.0;
}
}

/* This function is to transform the ring or BRW coordinates to the
```

```
   base coordinates with a combination of rotation and translation */
void TRANSFORM1(float T[3], float p1[3], float p2[3])
{
 int i, j;
 float c1, s1, c2, s2, c3, s3;
 float t_matrix[3][3];
 float center[3];

  for(j = 0; j < 3; j++) {
        center[j] = center2[j];
        p2[j] = 0.0;
  }

 c1 = cos(T[0]);
 s1 = sin(T[0]);
 c2 = cos(T[1]);
 s2 = sin(T[1]);
 c3 = cos(T[2]);
 s3 = sin(T[2]);

 for(i = 0; i < 3 ; i++) {
   for( j = 0; j < 3; j++)
    t_matrix[i][j] = 0.0;
 }

/* for(i = 0; i < 3; i++) {
        p1[i] -= center[i];
        printf("center2[%d]=%4.2f, p1[%d]=%4.2f\n", i, center2[i], i, p1[i]);
 } */

 // The followings are the Euler angles (Z-Y-X)
 t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = c1 * s2 * s3 - s1 * c3;
 t_matrix[0][2] = c1 * s2 * c3 + s1 * s3;
 t_matrix[1][0] = s1 * c2;
 t_matrix[1][1] = s1 * s2 * s3 + c1 * c3;
 t_matrix[1][2] = s1 * s2 * c3 - c1 * s3;
 t_matrix[2][0] = -s2;
 t_matrix[2][1] = c2 * s3;
 t_matrix[2][2] = c2 * c3;

 p2[0] = t_matrix[0][0] * p1[0] + t_matrix[0][1] * p1[1] +
         t_matrix[0][2] * p1[2];
 p2[1] = t_matrix[1][0] * p1[0] + t_matrix[1][1] * p1[1] +
         t_matrix[1][2] * p1[2];
 p2[2] = t_matrix[2][0] * p1[0] + t_matrix[2][1] * p1[1] +
         t_matrix[2][2] * p1[2];

/* for(i = 0; i < 3; i++) {
        p1[i] += center[i];
        printf("center2[%d]=%4.2f, p1[%d]=%4.2f\n", i, center2[i], i, p1[i]);
 }

 for(i = 0; i < 3; i++) {
        p2[i] += center[i];
        printf("center2[%d]=%4.2f, p2[%d]=%4.2f\n", i, center2[i], i, p2[i]);
 } */

}

/* This routine is to generate the object function for calculation
   It based on the minimization of vector differences of two data sets
   (In a least squared sense */
void obj_fun(T, f)
float T[3], *f;
{
 int i, j;
 float ep[N][3], delta[N];

 for(i = 0; i < numpoints; i++)
  delta[i] = 0.0;

 TRANSFORM(T,points_2r,points_2f);

 /* Calculate the vectors for 2nd day
   (Assume all vectors refer to the origin) */
 for(i = 0; i < numpoints; i++) {
  vector_2[i][0] = points_2f[i][0] ;
  vector_2[i][1] = points_2f[i][1] ;
  vector_2[i][2] = points_2f[i][2] ;
 }

 NORMAL(vector_2);   /* Normalize to the unitary function */

 for(i = 0; i < numpoints; i++) {
       delta[i] = 0.0;
  for(j = 0; j < 3; j++)
       ep[i][j] = vector_2[i][j] - vector_1[i][j];
  for(j = 0; j < 3; j++) {
        delta[i] += SQ(ep[i][j]);
  }
 }
  *f = 0.0;
 for(i = 0; i < numpoints; i++)
  *f += delta[i];
  *f = sqrt(*f);
}

/* Hooke - Jeeves pattern search optimization algorithm to solve
   the data points matching problem */
void HK_JV(t)
float t[3];
{
 static int cnt=0;
 int i, ID, IC;
 float f_min;
 float alph[3];
 float F, F1;
 float Th1[3],Th0[3];
 FILE *temp_output;

 temp_output = gfopen("hooke.dis", "a");

 ID = 0; IC = 1;

 /* 1st day angles as initial values */
 theta_2[0] = t[0];
 theta_2[1] = t[1];
 theta_2[2] = t[2];

 for (i = 0 ; i < 3 ; i++)
  alph[i] = 0.0001 ;  /* Set up the threshold for increment */

 obj_fun(theta_2,&f_min) ;

 STP1 :

 F = f_min;

 for (i = 0 ; i < 3 ; i++)
  Th1[i] = theta_2[i] ;
```

```
STP2 :

 printf("Iteration step count = %d !\n", ++cnt);
 fprintf(temp_output, "%d %f\n", cnt, F1);

 for (i = 0 ; i < 3 ; i++)
  {
   Th1[i] = Th1[i] + alph[i] ;
   obj_fun(Th1,&F1) ;

   if (F1 < F)
     F = F1 ;
   else
    {
     Th1[i] = Th1[i] - 2.0 * alph[i] ;

     obj_fun(Th1,&F1) ;

     if ( F1 < F)
      {
       F = F1 ;
       alph[i] = -alph[i] ; /* change search direction if the search direction fails */
      }
     else
     Th1[i] = Th1[i] + alph[i] ;  /* keep the original value */
    }
 }
if( F < f_min)
 {
  f_min = F ;
  for ( i = 0 ; i < 3 ; i++)
   {
    Th0[i] = theta_2[i] ;
    theta_2[i] = Th1[i] ;
    Th1[i] = 2.0 * Th1[i] - Th0[i] ;
   }
  obj_fun(Th1,&F1) ;

   if (F1 >= f_min)
   {
    for( i = 0 ; i < 3 ; i++)
     Th1[i] = theta_2[i] ;
   }
  else
```

```
    {
     ID = 1 ;
     goto STP2 ;
    }
  }
 if ( ID == 1)
  {
   ID = 0 ;
   IC = IC + 1 ;
   goto STP1 ;
  }
 obj_fun(Th1,&F1) ;

 for(i = 0; i < 3; i++)
  theta_2[i] = Th1[i];

  fclose(temp_output);
 }

/* Pirnt the output matrix to compare with Pelizzari's answer */
 void prn_matrix(angle, outfp)
 float angle[3];
 FILE *outfp;
 {
 int i, j;
 float c1, s1, c2, s2, c3, s3;
 float t_matrix[4][4];

 c1 = cos(angle[0]);
 s1 = sin(angle[0]);
 c2 = cos(angle[1]);
 s2 = sin(angle[1]);
 c3 = cos(angle[2]);
 s3 = sin(angle[2]);
 for(i = 0; i < 4 ; i++)
  {
   for( j = 0; j < 4; j++)
    t_matrix[i][j] = 0.0;
  }
/* User defined machanical system for rotation and translation */
/* t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = -c1 * s2 * c3 + s1 * s3;
 t_matrix[0][2] = c1 * s2 * s3 + s1 * c3;
 t_matrix[0][3] = c1 * c2 * T4 + s1 * (T2 + T3);
 t_matrix[1][0] = s2;
 t_matrix[1][1] = c2 * c3;
 t_matrix[1][2] = -c2 * s3;
 t_matrix[1][3] = s2 * T4;
 t_matrix[2][0] = -s1 * c2;
 t_matrix[2][1] = s1 * s2 * c3 + c1 * s3;
 t_matrix[2][2] = -s1 * s2 * s3 + c1 * c3;
 t_matrix[2][3] = -s1 * c2 * T4 + c1 * (T2 + T3) + T1;
 t_matrix[3][0] = 0.0;
 t_matrix[3][1] = 0.0;
 t_matrix[3][2] = 0.0;
 t_matrix[3][3] = 1.0;                */

 // The followings are the Euler angles (Z-Y-X) rotation, rotation sequence is important
!
 t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = c1 * s2 * s3 - s1 * c3;
 t_matrix[0][2] = c1 * s2 * c3 + s1 * s3;
 t_matrix[0][3] = c1 * c2 * (T1+T4) + c1 * s2 * (T3+T6) + c1 * T7
              - s1 * (T2+T5+T8);
 t_matrix[1][0] = s1 * c2;
 t_matrix[1][1] = s1 * s2 * s3 + c1 * c3;
 t_matrix[1][2] = s1 * s2 * c3 - c1 * s3;
 t_matrix[1][3] = s1 * c2 * (T1+T4) + s1 * s2 * (T3+T6) + s1 * T7
              + c1 * (T2+T5+T8);
 t_matrix[2][0] = -s2;
 t_matrix[2][1] = c2 * s3;
 t_matrix[2][2] = c2 * c3;
 t_matrix[2][3] = -s2 * (T1+T4) + c2 * (T3+T6) + T9;
 t_matrix[3][0] = 0.0;
 t_matrix[3][1] = 0.0;
 t_matrix[3][2] = 0.0;
 t_matrix[3][3] = 1.0;

 printf("The following is the rotation matrix !\n");
 for(i = 0; i < 4 ; i++) {
  for( j = 0; j < 4; j++)
   printf("%8.4f", t_matrix[i][j]);
  printf("\n");
 }

 fprintf(outfp, "The following is the rotation matrix !\n");
 for(i = 0; i < 4 ; i++) {
  for( j = 0; j < 4; j++)
```

```
     fprintf(outfp, "%8.4f", t_matrix[i][j]);
    fprintf(outfp,"\n");
  }
}

/* Calculate the translation vector for the two sets of data points
   and return them after being translated */
void TRANSLATION(data1, data2)
float data1[N][4], data2[N][4];
{
 int i, j;

 for(j = 0; j < 3; j++)
        center1[j] = center2[j] = 0.0;

 for(i = 0; i < numpoints; i++) {
   for(j = 0; j < 3; j++) {
        center1[j] += data1[i][j];
        center2[j] += data2[i][j];
   }
 }

 for(j = 0; j < 3; j++) {
        center1[j] /= numpoints;
        center2[j] /= numpoints;
        shift[j] = center1[j] - center2[j];
 }
}

/* Input and output files definition, grace function */
FILE *gfopen(char *fn, char *mode)
{
        FILE *fp;

        if ((fp = fopen(fn, mode)) == NULL) {
                fprintf(stderr, "Cannot open %s - bye !\n", fn);
                exit(1);
        }
        return(fp);
}
```

```
/**********************************************************************/
/*      SIMPLEX.C
This is a multidimensional optimization program based on the downhill
simplex method of Nelder and Mead (Press, Shoup & Mistree 1987).
**********************************************************************/
/* This program is the Downhill Simplex optimization algorithm
   for calculation of the six degree-of-freedom fitting
   parameters to realign the patient's fractionated treatment
   position.  This program requires the 3-D input data set which contains
   number of the total 3-D points (1st line) and the X-Y-Z coordinates
   of the two sets of points separated by space.  After calculation the
   parameters are returned either as a print out format or file format.
   The output file can be viewed from any kind of text editor.
   edited by Jack Yang at the Shands Cancer Center, Jan 15, 1994.
*/
/* Include all the Header Files */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

/* Dimensions of the appratus ( the true dimensions may vary ) */
/* The initial values are assumed zeros because of the Euler angle definition */
#define T1 0.0
#define T2 0.0
#define T3 0.0
#define T4 0.0
#define T5 0.0
#define T6 0.0
#define T7 0.0
#define T8 0.0
#define T9 0.0

#define N 400
#define SQ(x) ((x)*(x))
#define radtodeg 57.29578
#define degtorad 0.0174532

/* Below are the max number of function evaluations, reflection coefficient,
   contraction coefficient, and expansion coefficient.                  */
#define NMAX  5000
#define ALPHA 1.0
#define BETA  0.5
#define GAMMA 2.0
#define GET_PSUM for (j=0;j<ndim;j++) { for (i=0,sum=0.0;i<mpts;i++)\
                    sum += p[i][j]; psum[j]=sum;}

void amoeba(float p[4][3],float y[4],int ndim,float ftol,float (*funk)(float x[3]),
        int *nfunk);
float amotry(float p[4][3],float y[4],float psum[3],int ndim,float (*funk)(float x[3]),
         int ihi,int *nfunk,float fac);
float funk(float x[3]);
void nrerror(char error_text[]);
void TRANSLATION(float data1[N][4], float data2[N][4]);
void NORMAL(float A[N][3]);
void TRANSFORM(float T[3], float old[N][4], float new[N][4]);

/* Definition of global variables */
/*------------------------------------------------------------------*/
/* points_1r : 1st day, BRW or any specified coordinate */
/* points_1f : 1st day, fixed coordinate */
/* points_2r : 2nd day, BRW or any specified coordinate */
/* points_2f : 2nd day, fixed coordinate */
/* vector_1  : 1st day vectors to each point (in the fixed system) */
/* vector_2  : 2nd day vectors to each point (in the fixed system) */
/* offset    : parameter for calculating mean error, mean squared error and
               root mean squared error */
/* shift     : translation vector for patient realignment*/
/* center1   : The centroid of 1st point sets before transformation*/
/* center2   : The centroid of 2nd point sets after transformation*/
/*------------------------------------------------------------------*/
float points_1r[N][4], points_1f[N][4], points_2r[N][4], points_2f[N][4];
float vector_1[N][3], vector_2[N][3], theta_1[3], theta_2[3];
float offset[N][3], shift[3];
float center1[3], center2[3];
int numpoints = 0;
FILE *temp_output;

/* MAIN PROGRAM STARTS HERE */
void main(int argc, char *argv[])
{
  float p[4][3]={0,1,0,0,0,0,1,0,0,0,0,1},x[3],ftol=0.0001,y[4];
  int ndim=3,*nfunk,i,j;
  float norm[N],mean,mean_square,rms;
  FILE *fp,*outfp;

 if(argc != 3)
 {
   fprintf(stdout, "\n** You forgot to specify file names **\n");
```

```
    fprintf(stderr, "\n%s%s%s%s\n\n%s\n\n",
            "Usage:  ", argv[0], " input_file ", " output_file ",
            "(Ex: simplex [input file] [output file])" );
    exit(1);
   }

  fp = fopen(argv[1], "r+");
  outfp = fopen(argv[2], "w");
  temp_output = fopen("simplex.dis", "w");

  fscanf(fp, "%d", &numpoints);
        printf("Number of points = %d\n", numpoints);

  printf("\n ***** data sets ***** -- File: %s --",*(argv+1));
  for(i = 0; i < numpoints; i++) {
        points_1r[i][3] = 1.0;
        points_2r[i][3] = 1.0;
        fscanf(fp, "%f %f %f %f %f %f", &points_1r[i][0],
             &points_1r[i][1], &points_1r[i][2], &points_2r[i][0],
             &points_2r[i][1], &points_2r[i][2]);
  }

  printf("\n***** 1st day data *****       ***** 2nd day data *****\n");
  for(i = 0; i < numpoints; i++)
   printf("(%8.4f %8.4f %8.4f)  (%8.4f %8.4f %8.4f)\n", points_1r[i][0],
         points_1r[i][1], points_1r[i][2], points_2r[i][0], points_2r[i][1],
         points_2r[i][2]);

/* Now perform the translation of the two different data sets ---
   First, move the two point sets to the (0,0,0) of the ring system
   or any specified coordinate system.
   Second, return the calculated translation vector.
*/
 TRANSLATION(points_1r, points_2r);

 for(i = 0; i < numpoints; i++) {
   for(j = 0; j < 3; j++) {
        points_1r[i][j] -= center1[j];
        points_2r[i][j] -= center2[j];
   }
  }

 for(i = 0; i < 3; i++)
        theta_1[i] = 0;
 TRANSFORM(theta_1,points_1r,points_1f);

 /* calculate the vectors for 1st day
    (Assume all vectors refer to the origin)*/
 for(i = 0; i < numpoints; i++) {
  vector_1[i][0] = points_1f[i][0] ;
  vector_1[i][1] = points_1f[i][1] ;
  vector_1[i][2] = points_1f[i][2] ;
 }

 NORMAL(vector_1);

 for (i=0; i<ndim+1; i++) {
   for (j=0;j<ndim;j++)
        x[j] = p[i][j];

        y[i]=funk(x);
   printf("y[%d]=%8.4f\n", i, y[i]);  /* print out the initial calculated function values
*/
   }

/* The input vector y[] has components pre-initialized to the values of funk
   evaluated at the ndim+1 vertices (rows) of p[][].
*/

   amoeba(p,y,ndim,ftol,funk,nfunk);

   for (j=0;j<ndim;j++)
        theta_2[j]=p[0][j];

 /* Transform 2nd data sets to the fixed system w/ the calculated theta_2[3] */
 TRANSFORM(theta_2,points_2r,points_2f);

 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 3; j++) {
        points_1f[i][j] += center1[j];
        points_2f[i][j] += center2[j];
  }
 }

/* Re-calculate the translation vector of the
   least-square minimization point sets */

  TRANSLATION(points_1f, points_2f);
```

```
/* Add the translation vector of point sets #2 for statistical analysis */
 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 3; j++) {
        points_2f[i][j] += shift[j];
  }
 }

/*-----------------------------------------------------------------------*/
 printf("\n                    ***** Displacements for points *****");
 printf("\n(Longitudinal, Transverse, Vertical)");
 for(i = 0; i < numpoints; i++) {
  offset[i][0] = (points_2f[i][0] - points_1f[i][0]); /* in longitudinal direction */
  offset[i][1] = (points_2f[i][1] - points_1f[i][1]); /* in transverse direction */
  offset[i][2] = (points_2f[i][2] - points_1f[i][2]); /* in vertical direction */

 }

/* Statistical analysis */
/* Calculate the mean, mean_square, and rms of the data points */
  mean = mean_square = 0.0;
  for(i = 0; i < numpoints; i++) {
        norm[i] = 0.0;
    for(j = 0; j < 3; j++) {
        norm[i] += SQ(offset[i][j]);
      norm[i] = sqrt(norm[i]);
    }
  mean += norm[i];
  mean_square += SQ(norm[i]);
  }

  mean /= numpoints;
  mean_square /= numpoints;
  rms = sqrt(mean_square);


 printf("\n\n                    ***** RESULT *****");
 printf("\nDisplacement in longitudinal direction (X) : %8.3lf",shift[0]);
 printf("\nDisplacement in transverse direction   (Y) : %8.3lf",shift[1]);
 printf("\nDisplacement in vertivcal direction    (Z) : %8.3lf",shift[2]);
 printf("\nRotation of joint 4 (Euler definition: Z-Yaw)      : %8.3lf
deg",theta_2[0]*radtodeg);
 printf("\nRotation of joint 5 (Euler definition: Y-Pitch)    : %8.3lf
deg",theta_2[1]*radtodeg);
 printf("\nRotation of joint 6 (Euler definition: X-Roll)     : %8.3lf
deg\n",theta_2[2]*radtodeg);
 printf("\nmean  =  %8.3lf  mean_square  =  %8.3lf  rms  =  %8.3lf\n",  mean,
mean_square, rms);


/*------------------------------------------------------------------------*/
 fprintf(outfp, "\n*** 1st day data ***          *** 2nd day data ***          ***
Displacements for points ***\n");
 for(i = 0; i < numpoints; i++)
  fprintf(outfp, "(%8.4f %8.4f %8.4f)(%8.4f %8.4f %8.4f)(%8.4f %8.4f %8.4f)\n",
points_1f[i][0],
         points_1f[i][1], points_1f[i][2], points_2f[i][0], points_2f[i][1],
         points_2f[i][2], offset[i][0], offset[i][1], offset[i][2]);

 fprintf(outfp, "\n\ncenter1[0][1][2] = %8.4f %8.4f %8.4f\n", center1[0], center1[1],
center1[2]);
 fprintf(outfp, "\n\ncenter2[0][1][2] = %8.4f %8.4f %8.4f\n", center2[0], center2[1],
center2[2]);

 fprintf(outfp, "\n\n***** RESULT *****\n");
  fprintf(outfp, "Displacement in longitudinal direction (X) : %8.3lf\n",shift[0]);
  fprintf(outfp, "Displacement in transverse direction   (Y) : %8.3lf\n",shift[1]);
  fprintf(outfp, "Displacement in vertivcal direction    (Z) : %8.3lf\n",shift[2]);
  fprintf(outfp,  "\nRotation  of  joint  4  (Euler  definition:  Z-Yaw)     :  %8.3lf
deg",theta_2[0]*radtodeg);
  fprintf(outfp,  "\nRotation  of  joint  5  (Euler  definition:  Y-Pitch)  :  %8.3lf
deg",theta_2[1]*radtodeg);
  fprintf(outfp,  "\nRotation  of  joint  6  (Euler  definition:  X-Roll)    :  %8.3lf
deg\n",theta_2[2]*radtodeg);
  fprintf(outfp, "\nmean = %8.3lf mean_square = %8.3lf rms = %8.3lf\n", mean,
mean_square, rms);

  fclose(fp);
  fclose(outfp);
  fclose(temp_output);

}

/* The object fuction to be minimized */
float funk(float theta_2[3])
{
    int i, j;
    float ep[N][3], delta[N], f;
    for(i = 0; i < numpoints; i++)
            delta[i] = 0.0;

    TRANSFORM(theta_2,points_2r,points_2f);

  /* Calculate the vectors for 2nd day
    (Assume all vectors refer to the origin) */
    for(i = 0; i < numpoints; i++) {
          vector_2[i][0] = points_2f[i][0] ;
          vector_2[i][1] = points_2f[i][1] ;
          vector_2[i][2] = points_2f[i][2] ;
    }

    NORMAL(vector_2);   /* Normalize to the unitary function */

    for(i = 0; i < numpoints; i++) {
          delta[i] = 0.0;
    for(j = 0; j < 3; j++)
          ep[i][j] = vector_2[i][j] - vector_1[i][j];
    for(j = 0; j < 3; j++) {
          delta[i] += SQ(ep[i][j]);
   }
  }
          f = 0.0;
   for(i = 0; i < numpoints; i++)
          f += delta[i];
          f = sqrt(f);

/*  printf("f = %f\n", f); */
  return(f);
}

/* Function to normalize the vector */
void NORMAL(A)
float A[N][3];
{
 int i, j, k ;
 float s ;

 for(i = 0; i < numpoints; i++) {
  s = 0.0;
  for (j = 0; j < 3; j++)
    s += SQ(A[i][j]);
  for ( j = 0 ; j < 3 ; j++ )
```

```
   A[i][j] = A[i][j] / sqrt(s) ;
  }
 }

/* Transform the ring coordinates to the fixed coordinates */
void TRANSFORM(T, old, new)
float T[3], old[N][4], new[N][4];
{
 int i, j;
 float c1, s1, c2, s2, c3, s3;
 float t_matrix[4][4];
 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 4; j++)
  new[i][j] = 0.0;
 }
 c1 = cos(T[0]);
 s1 = sin(T[0]);
 c2 = cos(T[1]);
 s2 = sin(T[1]);
 c3 = cos(T[2]);
 s3 = sin(T[2]);
 for(i = 0; i < 4 ; i++)
  {
   for( j = 0; j < 4; j++)
    t_matrix[i][j] = 0.0;
  }
// The followings are the non-conventional settings from drawing
/* t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = -c1 * s2 * c3 + s1 * s3;
 t_matrix[0][2] = c1 * s2 * s3 + s1 * c3;
 t_matrix[0][3] = c1 * c2 * T4 + s1 * (T2 + T3);
 t_matrix[1][0] = s2;
 t_matrix[1][1] = c2 * c3;
 t_matrix[1][2] = -c2 * s3;
 t_matrix[1][3] = s2 * T4;
 t_matrix[2][0] = -s1 * c2;
 t_matrix[2][1] = s1 * s2 * c3 + c1 * s3;
 t_matrix[2][2] = -s1 * s2 * s3 + c1 * c3;
 t_matrix[2][3] = -s1 * c2 * T4 + c1 * (T2 + T3) + T1;
 t_matrix[3][0] = 0.0;
 t_matrix[3][1] = 0.0;
 t_matrix[3][2] = 0.0;
 t_matrix[3][3] = 1.0;                */
 /* The followings are the Euler angles (Z-Y-X) with
    different rotational origins */
 t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = c1 * s2 * s3 - s1 * c3;
 t_matrix[0][2] = c1 * s2 * c3 + s1 * s3;
 t_matrix[0][3] = c1 * c2 * (T1+T4) + c1 * s2 * (T3+T6) + c1 * T7
                - s1 * (T2+T5+T8);
 t_matrix[1][0] = s1 * c2;
 t_matrix[1][1] = s1 * s2 * s3 + c1 * c3;
 t_matrix[1][2] = s1 * s2 * c3 - c1 * s3;
 t_matrix[1][3] = s1 * c2 * (T1+T4) + s1 * s2 * (T3+T6) + s1 * T7
                + c1 * (T2+T5+T8);
 t_matrix[2][0] = -s2;
 t_matrix[2][1] = c2 * s3;
 t_matrix[2][2] = c2 * c3;
 t_matrix[2][3] = -s2 * (T1+T4) + c2 * (T3+T6) + T9;
 t_matrix[3][0] = 0.0;
 t_matrix[3][1] = 0.0;
 t_matrix[3][2] = 0.0;
 t_matrix[3][3] = 1.0;

 for(i = 0; i < numpoints; i++) {
 new[i][0] = t_matrix[0][0] * old[i][0] + t_matrix[0][1] * old[i][1] +
           t_matrix[0][2] * old[i][2] + t_matrix[0][3] ;
 new[i][1] = t_matrix[1][0] * old[i][0] + t_matrix[1][1] * old[i][1] +
           t_matrix[1][2] * old[i][2] + t_matrix[1][3] ;
 new[i][2] = t_matrix[2][0] * old[i][0] + t_matrix[2][1] * old[i][1] +
           t_matrix[2][2] * old[i][2] + t_matrix[2][3] ;
 new[i][3] = 1.0;
 }
}
/************************************************************************
****/
void amoeba(float p[4][3],float y[4],int ndim,float ftol,float (*funk)(float x[3]),
          int *nfunk)
  {
  int i,j,ilo,ihi,inhi,mpts=ndim+1;
  float ytry,ysave,sum,rtol,psum[3];
  float amotry(float p[4][3],float y[4],float psum[3],int ndim,
               float (*funk)(float x[3]),int ihi,int *nfunk,float fac);
  void nrerror(char error_text[]);

  *nfunk=0;
  GET_PSUM
```

```
/* First we determine which point is the highest (worst), next highest, and
   lowest (best) by looping over the points in the simplex.              */

   for (;;)
      {
      ilo=0;
      ihi=y[0]>y[1] ? (inhi=1,0) : (inhi=0,1);
      for (i=0;i<mpts;i++)
         {
         if (y[i] < y[ilo])
            {
            ilo=i;
            }
         if (y[i] > y[ihi])
            {
            inhi=ihi;
            ihi=i;
            }
         else if (y[i] > y[inhi])
            {
            if (i != ihi)
               {
               inhi=i;
               }
            }
         }
      rtol=2.0*fabs(y[ihi]-y[ilo])/(fabs(y[ihi])+fabs(y[ilo]));

/* Compute the fractional range from highest to lowest and return if ok.    */

      if (rtol < ftol) break;
      if (*nfunk >= NMAX) nrerror("Too many iterations in AMOEBA");

/* Begin a new iteration. First extrapolate by a factor ALPHA through the
   face of the simplex across from the high point. i.e. reflect the simplex
   from the high point.                                          */

      ytry=amotry(p,y,psum,ndim,funk,ihi,nfunk,-ALPHA);
      if (ytry <= y[ilo])

/* Gives a result better than the best point, so try an additional
   extrapolation by factor GAMMA.                                     */

         ytry=amotry(p,y,psum,ndim,funk,ihi,nfunk,GAMMA);
      else if (ytry >= y[inhi])

 /* The reflected point is worse than the second-highest, so look for an
    intermediate lower point. i.e. do a one dimensional contraction.        */

         {
         ysave=y[ihi];
         ytry=amotry(p,y,psum,ndim,funk,ihi,nfunk,BETA);
         if (ytry >= ysave)

 /* Can't seem to get rid of that high point. Better contract around the
    lowest (best) point.                                                   */

           {
           for (i=0;i<mpts;i++)
             {
             if (i != ilo)
                {
                for (j=0;j<ndim;j++)
                  {
                  psum[j]=0.5*(p[i][j]+p[ilo][j]);
                  p[i][j]=psum[j];
                  }
                y[i]=(*funk)(psum);
                }
             }
/* Keep track of function evaluations and recompute psum. */
             *nfunk += ndim;
             GET_PSUM
           }
         }
     }

     for (i=0;i<ndim+1;i++) {
        printf("y[%d]=%.3e     ", i, y[i]);
        for (j=0;j<ndim;j++)
              printf("p[%d][%d]=%.3f \t", i, j, p[i][j]);
              printf("\n");
     }

  }

/************************************************************************
****/
```

```
/* Extrapolates by a factor fac through the face of the simplex across from
   the high point, tries it, and replaces the high point if the new point is
   better.                                                         */

float amotry(float p[4][3],float y[4],float psum[3],int ndim,float (*funk)(float x[3]),
            int ihi,int *nfunk,float fac)
   {
   int j;
   float fac1,fac2,ytry,ptry[3];
   void nrerror(char error_text[]);

   fac1=(1.0-fac)/ndim;
   fac2=fac1-fac;
   for (j=0;j<ndim;j++) ptry[j]=psum[j]*fac1-p[ihi][j]*fac2;

/* Evaluate the function at the trial point. If it is better than the
   highest, then replace the highest.                                 */

   ytry=(*funk)(ptry);
   ++(*nfunk);
   if (ytry < y[ihi])
      {
      y[ihi]=ytry;
      for (j=0;j<ndim;j++)
         {
         psum[j] += ptry[j]-p[ihi][j];
         p[ihi][j]=ptry[j];
         }
      }
      fprintf(temp_output,"%d %f\n", *nfunk, ytry);
      return ytry;
   }

/*************************************************************************
****/
void nrerror(char error_text[])
   {
   fprintf(stderr,"Numerical Recipes run-time error...\n");
   fprintf(stderr,"%s\n",error_text);
   fprintf(stderr,"...now exiting to system...\n");
   exit(1);
   }

/* Calculate the translation vector for the two sets of data points
```

```
   and return them after being translated */
void TRANSLATION(data1, data2)
float data1[N][4], data2[N][4];
{
 int i, j;

 for(j = 0; j < 3; j++)
        center1[j] = center2[j] = 0.0;

 for(i = 0; i < numpoints; i++) {
   for(j = 0; j < 3; j++) {
        center1[j] += data1[i][j];
        center2[j] += data2[i][j];
   }
 }

 for(j = 0; j < 3; j++) {
        center1[j] /= numpoints;
        center2[j] /= numpoints;
        shift[j] = center1[j] - center2[j];
 }
}
```

```
/*************************************************************************/
/*      ANNEAL.C
This is a multidimensional optimization program based on the annealed
downhill simplex method of Nelder and Mead
*************************************************************************/
/* This program is the Downhill Simplex Annealing optimization algorithm
   for calculation of the six degree-of-freedom fitting
   parameters to realign the patient's fractionated treatment
   position.  This program requires the 3-D input data set which contains
   number of the total 3-D points (1st line) and the X-Y-Z coordinates
   of the two sets of points separated by space.  After calculation the
   parameters are returned either as a print out format or file format.
   The output file can be viewed from any kind of text editor.
   edited by Jack Yang at the Shands Cancer Center, Jan 15, 1994.
*/
/* Include all the Header Files */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

/* Dimensions of the appratus ( the true dimensions may vary ) */
/* The initial values are assumed zeros because of the Euler angle definition */
#define T1 0.0
#define T2 0.0
#define T3 0.0
#define T4 0.0
#define T5 0.0
#define T6 0.0
#define T7 0.0
#define T8 0.0
#define T9 0.0

#define N 400
#define SQ(x) ((x)*(x))
#define radtodeg 57.29578
#define degtorad 0.0174532

#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
```

```
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

/* Below are the max number of function evaluations, reflection coefficient,
   contraction coefficient, and expansion coefficient.                    */
#define ALPHA 1.0
#define BETA  0.5
#define GAMMA 2.0
#define GET_PSUM \
            for(n=0;n<ndim;n++) {for(sum=0.0,m=0;m<mpts;m++) \
            sum += p[m][n]; psum[n]=sum; }

void amoebsa(float p[4][3],float y[4],int ndim,float pb[3], float *yb, float ftol,
        float (*funk)(float x[3]),int *iter,float temptr);
float amotsa(float p[4][3],float y[4],float psum[3],int ndim,float pb[3],
        float *yb,float (*funk)(float x[3]),int ihi,float *yhi,float fac);
float funk(float x[3]);
void TRANSLATION(float data1[N][4], float data2[N][4]);
void NORMAL(float A[N][3]);
void TRANSFORM(float T[3], float old[N][4], float new[N][4]);
float ran1(long *idum);

/* Definition of global variables */
/*----------------------------------------------------------------*/
/* points_1r : 1st day, BRW or any specified coordinate */
/* points_1f : 1st day, fixed coordinate */
/* points_2r : 2nd day, BRW or any specified coordinate */
/* points_2f : 2nd day, fixed coordinate */
/* vector_1  : 1st day vectors to each point (in the fixed system) */
/* vector_2  : 2nd day vectors to each point (in the fixed system) */
/* offset    : parameter for calculating mean error, mean squared error and
               root mean squared error */
/* shift     : translation vector for patient realignment*/
/* center1   : The centroid of 1st point sets before transformation*/
/* center2   : The centroid of 2nd point sets after transformation*/
/*----------------------------------------------------------------*/
float points_1r[N][4], points_1f[N][4], points_2r[N][4], points_2f[N][4];
float vector_1[N][3], vector_2[N][3], theta_1[3], theta_2[3];
float offset[N][3], shift[3];
float center1[3], center2[3];
int numpoints = 0;
FILE *temp_output;

long idum=(-64); /* random seed definition */
```

```
float tt;

/* MAIN PROGRAM STARTS HERE */
void main(int argc, char *argv[])
{
   int i,iiter=20,iter,j,jiter,ndim=3,nit=0;
   float temptr=1.0e6,yb=1.0e30,ybb=1.0e30;
   float p[4][3]={0,1,0,0,0,0,1,0,0,0,0,1},x[3],ftol=1.0e-6,y[4],pb[3];
   /* p[][] defines the vertex of the simplex, ftol is the tolerance */
   float norm[N],mean,mean_square,rms;
   FILE *fp,*outfp;

  if(argc != 3)
  {
    fprintf(stdout, "\n** You forgot to specify file names **\n");
    fprintf(stderr, "\n%s%s%s%s\n\n%s\n\n",
            "Usage:  ", argv[0], " input_file ", " output_file ",
            "(Ex: anneal [input file] [output file])" );
    exit(1);
  }

 fp = fopen(argv[1], "r+");
 outfp = fopen(argv[2], "w");
 temp_output = fopen("anneal.dis", "w");

 fscanf(fp, "%d", &numpoints);
       printf("Number of points = %d\n", numpoints);

 printf("\n ***** data sets ***** -- File: %s --",*(argv+1));
 for(i = 0; i < numpoints; i++) {
       points_1r[i][3] = 1.0;
       points_2r[i][3] = 1.0;
       fscanf(fp, "%f %f %f %f %f %f", &points_1r[i][0],
            &points_1r[i][1], &points_1r[i][2], &points_2r[i][0],
            &points_2r[i][1], &points_2r[i][2]);
 }

 printf("\n***** 1st day data *****        ***** 2nd day data *****\n");
 for(i = 0; i < numpoints; i++)
  printf("(%8.4f %8.4f %8.4f)  (%8.4f %8.4f %8.4f)\n", points_1r[i][0],
       points_1r[i][1], points_1r[i][2], points_2r[i][0], points_2r[i][1],
       points_2r[i][2]);

/* Now perform the translation of the two different data sets ---
```

```
   First, move the two point sets to the (0,0,0) of the ring system
   or any specified coordinate system.
   Second, return the calculated translation vector.
 */

 TRANSLATION(points_1r, points_2r);

 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 3; j++) {
       points_1r[i][j] -= center1[j];
       points_2r[i][j] -= center2[j];
  }
 }

 for(i = 0; i < 3; i++)
       theta_1[i] = 0;

 TRANSFORM(theta_1,points_1r,points_1f);

 /* calculate the vectors for 1st day
    (Assume all vectors refer to the origin)*/
 for(i = 0; i < numpoints; i++) {
  vector_1[i][0] = points_1f[i][0] ;
  vector_1[i][1] = points_1f[i][1] ;
  vector_1[i][2] = points_1f[i][2] ;
 }

 NORMAL(vector_1);

  for (i=0; i<ndim+1; i++) {
   for (j=0;j<ndim;j++)
       x[j] = p[i][j];

   y[i] = funk(x);
   printf("y[%d]=%8.4f\n", i, y[i]);  /* print out the calculated function value y[] */
  }

/* The input vector y[] has components pre-initialized to the values of funk
   evaluated at the ndim+1 vertices (rows) of p[][].
*/

/* Peforming annealing optimization algorithm,
   for 150 iterations, results are OK and the process
   will be terminated.
```

```
*/
 for (jiter=0;jiter<150;jiter++) {
       iter=iiter;
       temptr *= 0.8;
       amoebsa(p,y,ndim,pb,&yb,ftol,funk,&iter,temptr);
       nit += iiter - iter;
       if (yb < ybb) {
               ybb=yb;
   /*          for (j=0;j<ndim;j++) printf("%10.5f",pb[j]); */
               printf("\n  ****yb****=%f\n",yb);
       }
     fprintf(temp_output,"%d %f\n", jiter+1, yb);
       printf("jiter # =%d\n", jiter);
       if(iter > 0) break;
 }

 printf("Vertices of fnal 3-D simplex and\n");
 printf("float values at the vertices:\n");
 printf("%3s %10s %12s %12s %14s\n\n","i","x[i]","y[i]","z[i]","function");
 for(i=0;i<ndim+1;i++) {
       printf("%3d",i);
       for(j=0;j<ndim;j++) printf("%12.6f",p[i][j]);
       printf("%15.7e\n",yb);
 }
 printf("%3d", 99);
 for(j=0;j<ndim;j++) printf("%12.6f",pb[j]);
 printf("%15.7e\n",yb);

/*******************************************/
  for (j=0;j<ndim;j++) {
    /* if(pb[j] <= -M_PI) pb[j] += M_PI;
       if(pb[j] <=-M_PI_2 && pb[j] > -M_PI) pb[j] += -M_PI;       */
       theta_2[j]=pb[j];
  }
 /* Transform 2nd data sets to the fixed system w/ the calculated theta_2[3] */

 TRANSFORM(theta_2,points_2r,points_2f);

 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 3; j++) {
       points_1f[i][j] += center1[j];
       points_2f[i][j] += center2[j];
  }
 }
```

```
/* Re-calculate the translation vector of the
   least-square minimization point sets */

  TRANSLATION(points_1f, points_2f);

/* Add the translation vector of point sets #2 for statistical analysis */
 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 3; j++) {
        points_2f[i][j] += shift[j];
  }
 }

/*------------------------------------------------------------------------*/
 printf("\n                   ***** Displacements for points *****");
 printf("\n(Longitudinal, Transverse, Vertical)");
 for(i = 0; i < numpoints; i++) {
  offset[i][0] = (points_2f[i][0] - points_1f[i][0]); /* in longitudinal direction */
  offset[i][1] = (points_2f[i][1] - points_1f[i][1]); /* in transverse direction */
  offset[i][2] = (points_2f[i][2] - points_1f[i][2]); /* in vertical direction */

 }

/* Statistical analysis */
/* Calculate the mean, mean_square, and rms of the data points */
  mean = mean_square = 0.0;
  for(i = 0; i < numpoints; i++) {
        norm[i] = 0.0;
    for(j = 0; j < 3; j++) {
        norm[i] += SQ(offset[i][j]);
      norm[i] = sqrt(norm[i]);
    }
  mean += norm[i];
  mean_square += SQ(norm[i]);
  }

  mean /= numpoints;
  mean_square /= numpoints;
  rms = sqrt(mean_square);


 printf("\n\n                   ***** RESULT *****");
 printf("\nDisplacement in longitudinal direction (X) : %8.3lf",shift[0]);
 printf("\nDisplacement in transverse direction   (Y) : %8.3lf",shift[1]);
 printf("\nDisplacement in vertivcal direction    (Z) : %8.3lf",shift[2]);
 printf("\nRotation of joint 4 (Euler definition: Z-Yaw)        : %8.3lf
deg",theta_2[0]*radtodeg);
 printf("\nRotation of joint 5 (Euler definition: Y-Pitch)  : %8.3lf
deg",theta_2[1]*radtodeg);
 printf("\nRotation of joint 6 (Euler definition: X-Roll)     : %8.3lf
deg\n",theta_2[2]*radtodeg);
 printf("\nmean = %8.3lf mean_square = %8.3lf rms = %8.3lf\n", mean,
mean_square, rms);


/*-----------------------------------------------------------------------*/
 fprintf(outfp, "\n*** 1st day data ***           *** 2nd day data ***           ***
Displacements for points ***\n");
 for(i = 0; i < numpoints; i++)
  fprintf(outfp, "(%8.4f %8.4f %8.4f)(%8.4f %8.4f %8.4f)(%8.4f %8.4f %8.4f)\n",
points_1f[i][0],
          points_1f[i][1], points_1f[i][2], points_2f[i][0], points_2f[i][1],
          points_2f[i][2], offset[i][0], offset[i][1], offset[i][2]);

 fprintf(outfp, "\n\ncenter1[0][1][2] = %8.4f %8.4f %8.4f\n", center1[0], center1[1],
center1[2]);
 fprintf(outfp, "\n\ncenter2[0][1][2] = %8.4f %8.4f %8.4f\n", center2[0], center2[1],
center2[2]);

 fprintf(outfp, "\n\n***** RESULT *****\n");
  fprintf(outfp, "Displacement in longitudinal direction (X) : %8.3lf\n",shift[0]);
  fprintf(outfp, "Displacement in transverse direction   (Y) : %8.3lf\n",shift[1]);
  fprintf(outfp, "Displacement in vertivcal direction    (Z) : %8.3lf\n",shift[2]);
  fprintf(outfp, "\nRotation of joint 4 (Euler definition: Z-Yaw)      : %8.3lf
deg",theta_2[0]*radtodeg);
  fprintf(outfp, "\nRotation of joint 5 (Euler definition: Y-Pitch) : %8.3lf
deg",theta_2[1]*radtodeg);
  fprintf(outfp, "\nRotation of joint 6 (Euler definition: X-Roll)    : %8.3lf
deg\n",theta_2[2]*radtodeg);
  fprintf(outfp, "\nmean = %8.3lf mean_square = %8.3lf rms = %8.3lf\n", mean,
mean_square, rms);

  fclose(fp);
  fclose(outfp);
  fclose(temp_output);

}

/* The fuction to be minimized */
```

```
float funk(float theta_2[3])
{
    int i, j;
    float ep[N][3], delta[N], f;

    for(i = 0; i < numpoints; i++)
            delta[i] = 0.0;

    TRANSFORM(theta_2,points_2r,points_2f);

 /* Calculate the vectors for 2nd day
   (Assume all vectors refer to the origin) */
   for(i = 0; i < numpoints; i++) {
         vector_2[i][0] = points_2f[i][0] ;
         vector_2[i][1] = points_2f[i][1] ;
         vector_2[i][2] = points_2f[i][2] ;
   }

   NORMAL(vector_2);   /* Normalize to the unitary function */

   for(i = 0; i < numpoints; i++) {
         delta[i] = 0.0;
   for(j = 0; j < 3; j++)
         ep[i][j] = vector_2[i][j] - vector_1[i][j];
   for(j = 0; j < 3; j++) {
         delta[i] += SQ(ep[i][j]);
  }
 }
         f = 0.0;
   for(i = 0; i < numpoints; i++)
         f += delta[i];
         f = sqrt(f);

 /* printf("%f\n", f);  */
  return(f);
}

/* Function to normalize the vector */
void NORMAL(A)
float A[N][3];
{
 int i, j, k ;
 float s ;
```

```
  for(i = 0; i < numpoints; i++) {
   s = 0.0;
   for (j = 0; j < 3; j++)
     s += SQ(A[i][j]);
   for ( j = 0 ; j < 3 ; j++ )
    A[i][j] = A[i][j] / sqrt(s) ;
  }
 }

/* Transform the ring coordinates to the fixed coordinates */
void TRANSFORM(T, old, new)
float T[3], old[N][4], new[N][4];
{
 int i, j;
 float c1, s1, c2, s2, c3, s3;
 float t_matrix[4][4];
 for(i = 0; i < numpoints; i++) {
  for(j = 0; j < 4; j++)
  new[i][j] = 0.0;
 }
 c1 = cos(T[0]);
 s1 = sin(T[0]);
 c2 = cos(T[1]);
 s2 = sin(T[1]);
 c3 = cos(T[2]);
 s3 = sin(T[2]);
 for(i = 0; i < 4 ; i++)
  {
   for( j = 0; j < 4; j++)
    t_matrix[i][j] = 0.0;
  }
// The followings are the non-conventional settings from drawing
/* t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = -c1 * s2 * c3 + s1 * s3;
 t_matrix[0][2] = c1 * s2 * s3 + s1 * c3;
 t_matrix[0][3] = c1 * c2 * T4 + s1 * (T2 + T3);
 t_matrix[1][0] = s2;
 t_matrix[1][1] = c2 * c3;
 t_matrix[1][2] = -c2 * s3;
 t_matrix[1][3] = s2 * T4;
 t_matrix[2][0] = -s1 * c2;
 t_matrix[2][1] = s1 * s2 * c3 + c1 * s3;
 t_matrix[2][2] = -s1 * s2 * s3 + c1 * c3;
 t_matrix[2][3] = -s1 * c2 * T4 + c1 * (T2 + T3) + T1;
```

```
t_matrix[3][0] = 0.0;
t_matrix[3][1] = 0.0;
t_matrix[3][2] = 0.0;
t_matrix[3][3] = 1.0;          */

/* The followings are the Euler angles (Z-Y-X) with
   different rotational origins */
t_matrix[0][0] = c1 * c2;
t_matrix[0][1] = c1 * s2 * s3 - s1 * c3;
t_matrix[0][2] = c1 * s2 * c3 + s1 * s3;
t_matrix[0][3] = c1 * c2 * (T1+T4) + c1 * s2 * (T3+T6) + c1 * T7
               - s1 * (T2+T5+T8);
t_matrix[1][0] = s1 * c2;
t_matrix[1][1] = s1 * s2 * s3 + c1 * c3;
t_matrix[1][2] = s1 * s2 * c3 - c1 * s3;
t_matrix[1][3] = s1 * c2 * (T1+T4) + s1 * s2 * (T3+T6) + s1 * T7
               + c1 * (T2+T5+T8);
t_matrix[2][0] = -s2;
t_matrix[2][1] = c2 * s3;
t_matrix[2][2] = c2 * c3;
t_matrix[2][3] = -s2 * (T1+T4) + c2 * (T3+T6) + T9;
t_matrix[3][0] = 0.0;
t_matrix[3][1] = 0.0;
t_matrix[3][2] = 0.0;
t_matrix[3][3] = 1.0;

for(i = 0; i < numpoints; i++) {
new[i][0] = t_matrix[0][0] * old[i][0] + t_matrix[0][1] * old[i][1] +
          t_matrix[0][2] * old[i][2] + t_matrix[0][3] ;
new[i][1] = t_matrix[1][0] * old[i][0] + t_matrix[1][1] * old[i][1] +
          t_matrix[1][2] * old[i][2] + t_matrix[1][3] ;
new[i][2] = t_matrix[2][0] * old[i][0] + t_matrix[2][1] * old[i][1] +
          t_matrix[2][2] * old[i][2] + t_matrix[2][3] ;
new[i][3] = 1.0;
}
}
/**********************************************************************
****/
void amoebsa(float p[4][3],float y[4],int ndim,float pb[3], float *yb, float ftol,
          float (*funk)(float x[3]),int *iter,float temptr)
{
  int i,j,m,n,ilo,ihi,mpts=ndim+1;
  float rtol,sum,swap,yhi,ylo,ynhi,ysave,yt,ytry,psum[3];
  float tt;
```

```
   tt = -temptr;
   GET_PSUM

/* First we determine which point is the highest (worst), next highest, and
   lowest (best) by looping over the points in the simplex.            */

   for (;;) {
      ilo=0;
      ihi=1;
      ynhi=ylo=y[0]+tt*(ran1(&idum));
      yhi=y[1]+tt*log(ran1(&idum));

      if (ylo > yhi) {
         ihi=0;
         ilo=1;
         ynhi=yhi;
         yhi=ylo;
         ylo=ynhi;
      }
      for (i=2;i<mpts;i++) {
         yt=y[i]+tt*log(ran1(&idum));
          if (yt <= ylo) {
             ilo=i;
             ylo=yt;
          }
          if (yt > yhi) {
             ynhi=yhi;
             ihi=i;
             yhi=yt;
          } else if (yt > ynhi) {
             ynhi=yt;
          }
      }
      rtol=2.0*fabs(yhi-ylo)/(fabs(yhi)+fabs(ylo));

/* Compute the fractional range from highest to lowest and return if ok.     */
/* If returning, put the best point in slot 0 */
      if (rtol < ftol || *iter < 0) {
         swap=y[0];
         y[0]=y[ilo];
         y[ilo]=swap;
         for (n=0;n<ndim;n++) {
                swap=p[0][n];
                p[0][n]=p[ilo][n];
```

```
                    p[ilo][n]=swap;
              }
  /* printf("***rtol*** = %f, ***iter*** = %d\n", rtol,*iter); */
              break;
          }
          *iter -= 2;

/* Begin a new iteration. First extrapolate by a factor ALPHA through the
   face of the simplex across from the high point. i.e. reflect the simplex
   from the high point.                                              */

          ytry=amotsa(p,y,psum,ndim,pb,yb,funk,ihi,&yhi,-ALPHA);
          if (ytry <= ylo) {

/* Gives a result better than the best point, so try an additional
   extrapolation by factor GAMMA.                                         */

          ytry=amotsa(p,y,psum,ndim,pb,yb,funk,ihi,&yhi,GAMMA); }
          else if (ytry >= ynhi)

/* The reflected point is worse than the second-highest, so look for an
   intermediate lower point. i.e. do a one dimensional contraction.       */

             {
             ysave=yhi;
             ytry=amotsa(p,y,psum,ndim,pb,yb,funk,ihi,&yhi,BETA);
             if (ytry >= ysave) {

/* Can't seem to get rid of that high point. Better contract around the
   lowest (best) point.                                              */

                for (i=0;i<mpts;i++) {
                  if (i != ilo) {
                     for (j=0;j<ndim;j++) {
                        psum[j]=0.5*(p[i][j]+p[ilo][j]);
                        p[i][j]=psum[j];
                     }
                     y[i]=(*funk)(psum);
                  }
                }
                *iter -= ndim;
                GET_PSUM       /* Keep track of function evaluations and recompute psum.
                   */
                }
```

```
        } else ++(*iter);
    }

/*      for (i=0;i<ndim+1;i++) {
         printf("y[i]=%.3e     ", y[i]);
         for (j=0;j<ndim;j++)
                printf("p=%.3f \t", p[i][j]);
                printf("\n");
      }             */

}

/*********************************************************************
****/

/* Extrapolates by a factor fac through the face of the simplex across from
   the high point, tries it, and replaces the high point if the new point is
   better.                                                         */

float amotsa(float p[4][3],float y[4],float psum[3],int ndim,float pb[3],
            float *yb,float (*funk)(float x[3]),int ihi,float *yhi,float fac)
{
   int j;
   float fac1,fac2,yflu,ytry,ptry[3];

   fac1=(1.0-fac)/ndim;
   fac2=fac1-fac;
   for (j=0;j<ndim;j++) ptry[j]=psum[j]*fac1-p[ihi][j]*fac2;

/* Evaluate the function at the trial point. If it is better than the
   highest, then replace the highest.                                */

   ytry=(*funk)(ptry);

   if (ytry <= *yb) {
     for (j=0;j<ndim;j++) pb[j]=ptry[j];
     *yb=ytry;
   }
   yflu=ytry-tt*(ran1(&idum));
   if (yflu < *yhi) {
       y[ihi]=ytry;
       *yhi=yflu;
       for (j=0;j<ndim;j++) {
             psum[j] += ptry[j]-p[ihi][j];
```

```
                p[ihi][j]=ptry[j];
        }
    }
    return yflu;
}

/* Calculate the translation vector for the two sets of data points
   and return them after being translated */
void TRANSLATION(data1, data2)
float data1[N][4], data2[N][4];
{
 int i, j;

 for(j = 0; j < 3; j++)
        center1[j] = center2[j] = 0.0;

 for(i = 0; i < numpoints; i++) {
   for(j = 0; j < 3; j++) {
        center1[j] += data1[i][j];
        center2[j] += data2[i][j];
   }
 }

 for(j = 0; j < 3; j++) {
        center1[j] /= numpoints;
        center2[j] /= numpoints;
        shift[j] = center1[j] - center2[j];
 }
}

/* This routine provides the random number generator for the
   remodification of the simplex method, also provide the groundwork
   for annealing procedures */
float ran1(long *idum)
{
        int j;
        long k;
        static long iy=0;
        static long iv[NTAB];
        float temp;

        if (*idum <= 0 || !iy) {
                if (-(*idum) < 1) *idum=1;
                else *idum = -(*idum);
        for (j=NTAB+7;j>=0;j--) {
            k=(*idum)/IQ;
            *idum=IA*(*idum-k*IQ)-IR*k;
            if (*idum < 0) *idum += IM;
            if (j < NTAB) iv[j] = *idum;
        }
        iy=iv[0];
    }
    k=(*idum)/IQ;
    *idum=IA*(*idum-k*IQ)-IR*k;
    if (*idum < 0) *idum += IM;
    j=iy/NDIV;
    iy=iv[j];
    iv[j] = *idum;
    if ((temp=AM*iy) > RNMX) return RNMX;
    else return temp;
}
```

```
/***************************************************************************/
/*      RANDOM.C
This is a random process program whose function is to generate 100 sets
of randomly translated, rotated and with Gaussian noise data sets.  These
data sets are then used to simulate the inverse cauculation of the SVD and
optimization programs described in CHapter 6      */
/***************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>

/* dimensions of the appratus */
#define T1 0.0
#define T2 0.0
#define T3 0.0
#define T4 0.0
#define radtodeg 57.29578
#define degtorad 0.0174532

#define M1 259200
#define IA1 7141
#define IC1 54773
#define RM1 (1.0/M1)
#define M2 134456
#define IA2 8121
#define IC2 28411
#define RM2 (1.0/M2)
#define M3 243000
#define IA3 4561
#define IC3 51349

#define points 20
int idum = -100;

/* Function prototypes */
void TRANSFORM(float theta[3], float dist[3],
               float input[points][4], float output[points][4]);
float gasdev(int *idum);
float ran1(int *idum);
FILE *gfopen(char *fn, char *mode);

void main(int argc, char *argv[])
```

```
{
 int i, j;
 float input[points][4], output[points][4];
 float theta[3], trans[3];
 FILE *fp, *outfp;

 clrscr();
 fprintf(stdout, " Calculate random point sets after some \
                  rotation and translation !\n");
 if(argc != 3)
  {
   fprintf(stdout, "\n** You forgot to specify file name **\n");
   fprintf(stderr, "\n%s%s%s%s\n\n%s\n\n",
           "Usage:  ", argv[0], " input_file ", " output_file ",
           "(Ex: random [input.dat] [output.dat])" );
   exit(1);
  }
 fp = gfopen(argv[1], "r+");
 outfp = gfopen(argv[2], "w");

 for(i = 0; i < points; i++) {
       for(j = 0; j < 3; j++)
              input[i][j] = 0.0;
 }
 for(i = 0; i < points; i++) {
       input[i][3] = 1.0;
       input[i][3] = 1.0;
       fscanf(fp, "%f %f %f", &input[i][0],
             &input[i][1], &input[i][2]);
 }

/* Install the random generator */
/* ran1 returns a uniform random deviate between 0.0 and 1.0 */
/* theta is between -20 and 20 degrees */
/* trans is between than -50 and 50 mm */
 for( i = 0; i < 3; i++) {
       theta[i] = 0.35*(ran1(&idum)-1);
       trans[i] = 50*(ran1(&idum)-1);
 }
 /* Transform the data set by rotation and translation */
 TRANSFORM(theta, trans, input, output);

 /* Add Gaussian noises to the rotated and translated data sets */
 /* gasdev() returns normally distributed deviate with zero mean and
```

```
    unit variance */
 for(i = 0; i < points; i++) {
        for(j = 0; j < 3; j++) {
                output[i][j] += gasdev(&idum);
        }
 }
 for(i = 0; i < points; i++)
        fprintf(outfp, "%f %f %f %f %f %f\n", input[i][0], input[i][1],
                input[i][2], output[i][0], output[i][1], output[i][2]);
 fprintf(outfp, "\n\nThe following are random rotation angles and translation
distances\n");

 for(i = 0; i < 3; i++)
        fprintf(outfp, "Theta[%d] = %f\n", i, theta[i]);
 for(i = 0; i < 3; i++)
        fprintf(outfp, "Dist[%d] = %f\n", i, trans[i]);
}

void TRANSFORM(angle, distance, old, new)
float angle[3], distance[3], old[points][4], new[points][4];
{
 int i, j;
 float c1, s1, c2, s2, c3, s3;
 float t_matrix[4][4];
 for(i = 0; i < points; i++) {
  for(j = 0; j < 4; j++)
  new[i][j] = 0.0;
 }
 c1 = cos(angle[0]);
 s1 = sin(angle[0]);
 c2 = cos(angle[1]);
 s2 = sin(angle[1]);
 c3 = cos(angle[2]);
 s3 = sin(angle[2]);
 for(i = 0; i < 4 ; i++)
  {
   for( j = 0; j < 4; j++)
    t_matrix[i][j] = 0.0;
  }
/* t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = -c1 * s2 * c3 + s1 * s3;
 t_matrix[0][2] = c1 * s2 * s3 + s1 * c3;
 t_matrix[0][3] = c1 * c2 * T4 + s1 * (T2 + T3) + distance[0];
 t_matrix[1][0] = s2;
 t_matrix[1][1] = c2 * c3;
 t_matrix[1][2] = -c2 * s3;
 t_matrix[1][3] = s2 * T4 + distance[1];
 t_matrix[2][0] = -s1 * c2;
 t_matrix[2][1] = s1 * s2 * c3 + c1 * s3;
 t_matrix[2][2] = -s1 * s2 * s3 + c1 * c3;
 t_matrix[2][3] = -s1 * c2 * T4 + c1 * (T2 + T3) + T1 + distance[2];
 t_matrix[3][0] = 0.0;
 t_matrix[3][1] = 0.0;
 t_matrix[3][2] = 0.0;
 t_matrix[3][3] = 1.0;                    */

 // The followings are the Euler angles (Z-Y-X)
 t_matrix[0][0] = c1 * c2;
 t_matrix[0][1] = c1 * s2 * s3 - s1 * c3;
 t_matrix[0][2] = c1 * s2 * c3 + s1 * s3;
 t_matrix[0][3] = distance[0];
 t_matrix[1][0] = s1 * c2;
 t_matrix[1][1] = s1 * s2 * s3 + c1 * c3;
 t_matrix[1][2] = s1 * s2 * c3 - c1 * s3;
 t_matrix[1][3] = distance[1];
 t_matrix[2][0] = -s2;
 t_matrix[2][1] = c2 * s3;
 t_matrix[2][2] = c2 * c3;
 t_matrix[2][3] = distance[2];
 t_matrix[3][0] = 0.0;
 t_matrix[3][1] = 0.0;
 t_matrix[3][2] = 0.0;
 t_matrix[3][3] = 1.0;

 for(i = 0; i < points; i++) {
 new[i][0] = t_matrix[0][0] * old[i][0] + t_matrix[0][1] * old[i][1] +
             t_matrix[0][2] * old[i][2] + t_matrix[0][3] ;
 new[i][1] = t_matrix[1][0] * old[i][0] + t_matrix[1][1] * old[i][1] +
             t_matrix[1][2] * old[i][2] + t_matrix[1][3] ;
 new[i][2] = t_matrix[2][0] * old[i][0] + t_matrix[2][1] * old[i][1] +
             t_matrix[2][2] * old[i][2] + t_matrix[2][3] ;
 new[i][3] = 1.0;
 }
}

float gasdev(int *idum)
{
 static int iset=0;
 static float gset;
 float fac, r, v1,v2;
 float ran1();

 if (iset == 0) {
        do {
                v1 = 2.0*ran1(idum)-1.0;
                v2 = 2.0*ran1(idum)-1.0;
                r = v1*v1+v2*v2;
        } while(r >= 1.0 || r == 0.0);
        fac = sqrt(-2.0*log(r)/r);
        gset = v1*fac;
        iset = 1;
        return(v2*fac);
 }
 else {
        iset = 0;
        return gset;
 }
}
float ran1(int *idum)
{
 static long ix1, ix2, ix3;
 static float r[98];
 float temp;
 static int iff=0;
 int j;

 if(*idum < 0 || iff ==0) {
        iff = 1;
        ix1 = (IC1-(*idum)) % M1;
        ix1 = (IA1*ix1+IC1) % M1;
        ix2 = ix1 % M2;
        ix1 = (IA1*ix1+IC1) % M1;
        ix3 = ix1 % M3;
        for (j=1; j<97; j++) {
                ix1 = (IA1*ix1+IC1) % M1;
                ix2 = (IA2*ix2+IC2) % M2;
                r[j] = (ix1+ix2*RM2)*RM1;
        }
        *idum = 1;
}
ix1 = (IA1*ix1+IC1) % M1;
ix2 = (IA2*ix2+IC2) % M2;
```

```
 ix3 = (IA3*ix3+IC3) % M3;
 j = 1+((97*ix3)/M3);
 if (j>97 || j<1) printf("RAN1: This cannot happen");
 temp=r[j];
 r[j] = (ix1+ix2*RM2)*RM1;
 return(temp);
}

FILE *gfopen(char *fn, char *mode)
{
        FILE *fp;

        if ((fp = fopen(fn, mode)) == NULL) {
                fprintf(stderr, "Cannot open %s - bye !\n", fn);
                exit(1);
        }
       return(fp);
}
```

```
/*      FILTER.C      */
/* This program is to sort out the contour line which was obtained from the thresholding
   technique in the new SRS image localization porogram.  Because the drag box defines
   the contour line from a top to bottom and left to right sequence.  This program helps
   to sort out the contour line points in an increasing axial coordinates.  Therefore, on
   a sagittal cut plane the line should be from right to the left in sequance.
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <malloc.h>

/* Function prototypes */

FILE *gfopen(char *fn, char *mode);
void sort3(int n, float ra[], float rb[], float rc[]);
void indexx(int n, float arrin[], int indx[]);
int *ivector(int nl, int nh);
float *vector(int nl, int nh);
void free_ivector(int *v, int nl, int nh);
void free_vector(float *v, int nl, int nh);

/* MAIN PROGRAM */
void main(int argc, char *argv[])
{
        int i, contour1_num=0, contour2_num=0;
        float *contour1_pts_x, *contour1_pts_y, *contour1_pts_z;
        float *contour2_pts_x, *contour2_pts_y, *contour2_pts_z;
        float tempx, tempy, tempz;
        float temp1, temp2, temp3;
        FILE *fp1, *fp2, *outfp1, *outfp2;


        if(argc != 5)
        {
                fprintf(stdout, "\n** You forgot to specify file names **\n");
                fprintf(stderr, "\n%s%s%s%s%s%s\n\n%s\n\n",
                "Usage:   ", argv[0], " input_file1 ", "inout_file2", " output_file1 ",
"output_file2",
                "(Ex: filter [contour1] [contour2] [out_contour1] [out_contour2)" );
                exit(0);
        }
        fp1 = gfopen(argv[1], "r+");
        fp2 = gfopen(argv[2], "r+");
        outfp1 = gfopen(argv[3], "w");
        outfp2 = gfopen(argv[4], "w");

/* Comment out this part for command line input
        fp1 = gfopen("data1-2", "r+");
        fp2 = gfopen("data2-2", "r+");
        outfp1 = gfopen("sort1", "w");
        outfp2 = gfopen("sort2", "w");      */

 /* Scan the contour file to see how many data points are there */
 while(fscanf(fp1, "%f %f %f", &tempx, &tempy, &tempz) != EOF)
        contour1_num++;

 while(fscanf(fp2, "%f %f %f", &tempx, &tempy, &tempz) != EOF)
        contour2_num++;

        printf("contour_1=%d contour_2=%d\n", contour1_num, contour2_num);
   /*    print out the number of points
         which are formed by either the CT or digitized data */

 rewind(fp1); /* rewind the data file */
 rewind(fp2);

 /* Allocate the memory size of the contour points */
        contour1_pts_x = vector(1,contour1_num);
        contour1_pts_y = vector(1,contour1_num);
        contour1_pts_z = vector(1,contour1_num);
        contour2_pts_x = vector(1,contour2_num);
        contour2_pts_y = vector(1,contour2_num);
        contour2_pts_z = vector(1,contour2_num);

 /* Scan the data files and save into buffer */
 /* Data file # 1 */
 for (i=1; i<=contour1_num; i++) {
        fscanf(fp1, "%f%f%f", &temp1, &temp2, &temp3);
         contour1_pts_x[i] = temp1;
         contour1_pts_y[i] = temp2;
         contour1_pts_z[i] = temp3;

        printf("%f %f %f\n", contour1_pts_x[i],
              contour1_pts_y[i], contour1_pts_z[i]);
 }
```

```
/* Data file # 2 */
 for (i=1; i<=contour2_num; i++) {
        fscanf(fp2, "%f%f%f", &temp3, &temp2, &temp1);
         contour2_pts_x[i] = temp1;
         contour2_pts_y[i] = temp2;
         contour2_pts_z[i] = temp3;

        printf("%f %f %f\n", contour2_pts_x[i],
              contour2_pts_y[i], contour2_pts_z[i]);
 }

 /* Start to sort to generate the sort1 and sort2 data, the data files can be defined by
users */
        sort3(contour1_num, contour1_pts_z, contour1_pts_y, contour1_pts_x);

  /* Modify the ouput for the CT data because of the shift in the scanning coordinates
*/

        for(i=1; i<=contour1_num; i++)
               fprintf(outfp1, "%f %f %f\n", contour1_pts_x[i] + 96.68,
               contour1_pts_y[i] - 9.50, contour1_pts_z[i] + 46.03);


        sort3(contour2_num, contour2_pts_z, contour2_pts_y, contour2_pts_x);

        for(i=1; i<=contour2_num; i++)
               fprintf(outfp2, "%f %f %f\n", contour2_pts_x[i],
               contour2_pts_y[i], contour2_pts_z[i]);

               printf("This is the end of program !");

 free_vector(contour1_pts_x,1,contour1_num);
 free_vector(contour1_pts_y,1,contour1_num);
 free_vector(contour1_pts_z,1,contour1_num);
 free_vector(contour2_pts_x,1,contour2_num);
 free_vector(contour2_pts_y,1,contour2_num);
 free_vector(contour2_pts_z,1,contour2_num);

 fclose(fp1);
 fclose(fp2);
 fclose(outfp1);
 fclose(outfp2);
}
```

```
/* Input and output files definition */
FILE *gfopen(char *fn, char *mode)
{
        FILE *fp;

        if ((fp = fopen(fn, mode)) == NULL) {
                fprintf(stderr, "Cannot open %s - bye !\n", fn);
                exit(1);
        }
        return(fp);
}

/* Heapsort algorithm to re-sort the contour data and indexing, this is the most stable
method
  in sorting the coordinates in certain dimension
*/

void sort3(int n, float ra[], float rb[], float rc[])
{
        int j, *iwksp, *ivector();
        float *wksp, *vector();
        void indexx(), free_vector(), free_ivector();

        iwksp = ivector(1,n);
        wksp = vector(1,n);

        indexx(n,ra,iwksp);
        for (j=1; j<=n; j++) wksp[j]=ra[j];
        for (j=1; j<=n; j++) ra[j] = wksp[iwksp[j]];
        for (j=1; j<=n; j++) wksp[j]=rb[j];
        for (j=1; j<=n; j++) rb[j] = wksp[iwksp[j]];
        for (j=1; j<=n; j++) wksp[j]=rc[j];
        for (j=1; j<=n; j++) rc[j] = wksp[iwksp[j]];

        free_ivector(iwksp,1,n);
        free_vector(wksp,1,n);

}
/* routine for indexing */
void indexx(int n, float arrin[], int indx[])
{
        int l, j, ir, indxt, i;
        float q;
        for(j=1; j<=n; j++) indx[j] = j;
        if(n == 1) return;
        l=(n >> 1) + 1;
        ir=n;
        for(;;) {
                if (l>1)
                        q=arrin[(indxt=indx[--l])];
                else {
                        q=arrin[(indxt=indx[ir])];
                        indx[ir]=indx[1];
                        if (--ir == 1) {
                                indx[1] = indxt;
                                return;
                        }
                }
                i=l;
                j=l << 1;
                while (j <= ir) {
                        if (j < ir && arrin[indx[j]] < arrin[indx[j+1]]) j++;
                        if (q < arrin[indx[j]]) {
                                indx[i] = indx[j];
                                j += (i=j);
                        }
                        else j=ir+1;
                }
                indx[i] = indxt;
        }
}

int *ivector(int nl, int nh)
{
        int *v;

        v=(int *) malloc((unsigned) (nh-nl+1)*sizeof(int));
        if (!v)  printf("Allocation error in ivector() !");
        return v-nl;
}

float *vector(int nl, int nh)
{
        float *v;

        v=(float *) malloc((unsigned) (nh-nl+1)*sizeof(float));
        if (!v)  printf("Allocation error in vector() !");
        return v-nl;
}

void free_ivector(int *v, int nl, int nh)
{
        free((char *) (v+nl));
}

void free_vector(float *v, int nl, int nh)
{
        free((char *) (v+nl));
}
```

```
/*      SPLINE.C     */
/* This program is to use path length concept to resample the
   3-D spline fitted curves of the CT/MR dataset as well as the digitized
   data set. The reference could be seen in the Wofgang's paper. Before using
   this program, the 3-D coordinates of the initial contour points have to
   be reordered by the filter program.
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define SQ(x) ((x)*(x))

/* Function prototypes */

FILE *gfopen(char *fn, char *mode); /* Grace open file routine */
void ispline_gen(float *X1, float *Y1, int len1,
                 float *X2, float *Y2, int len2,
                 float *dyofdx, float *d2yofdx2);
void getYD_gen(float *X, float *Y, float *YD, int len);
void tridiag_gen(float *A, float *B, float *C, float *D, int len);
void swap(float a[],int i, int j);

/* MAIN PROGRAM */

void main(int argc, char *argv[])
{
 int i, j, k;

 /* Input contour points of curve 1 and curve 2 */
 float *contour1_pts_x, *contour1_pts_y, *contour1_pts_z;
 float *contour2_pts_x, *contour2_pts_y, *contour2_pts_z;
 int contour1_num=0, contour2_num=0, min;
 float *path_length_1, *path_length_2; /* Original path length definition  of the curve */

 /* Output contour points of curve 1 and curve 2 with spline function */
 float *path_length_1_spline, *path_length_2_spline;
 float *contour1_spline_x, *contour1_spline_y, *contour1_spline_z;
 float *contour2_spline_x, *contour2_spline_y, *contour2_spline_z;
 float *d1_x, *d1_y, *d1_z, *d2_x, *d2_y, *d2_z;
 float *dd1_x, *dd1_y, *dd1_z, *dd2_x, *dd2_y, *dd2_z;
 float K1, K2, K3, K, S, *curvature1, *curvature2;
 /* The numbers of old and new contour points definition */
 int contour1_num_spline=200, contour2_num_spline=200; /* The specified number of
resample spline points */

 FILE *fp1, *fp2, *outfp1, *outfp2, *path1_fp, *path2_fp;
 float temp1, temp2, temp3;
 float x_difference; /* X directional difference between the maximum curvature points,
not used if
                          the same starting and ending points of contour lines are selected  */
 int index_1, index_2;

 float tempx, tempy, tempz;

/* Memory allocation process */

 path_length_1_spline = (float *) calloc(contour1_num_spline,sizeof(float));
 path_length_2_spline = (float *) calloc(contour2_num_spline,sizeof(float));
 contour1_spline_x = (float *) calloc(contour1_num_spline,sizeof(float));
 contour1_spline_y = (float *) calloc(contour1_num_spline,sizeof(float));
 contour1_spline_z = (float *) calloc(contour1_num_spline,sizeof(float));
 contour2_spline_x = (float *) calloc(contour2_num_spline,sizeof(float));
 contour2_spline_y = (float *) calloc(contour2_num_spline,sizeof(float));
 contour2_spline_z = (float *) calloc(contour2_num_spline,sizeof(float));

/* First and second order derivatives of the two splined fitted curves */
 d1_x = (float *) calloc(contour1_num_spline,sizeof(float));
 d1_y = (float *) calloc(contour1_num_spline,sizeof(float));
 d1_z = (float *) calloc(contour1_num_spline,sizeof(float));
 dd1_x = (float *) calloc(contour1_num_spline,sizeof(float));
 dd1_y = (float *) calloc(contour1_num_spline,sizeof(float));
 dd1_z = (float *) calloc(contour1_num_spline,sizeof(float));
 curvature1 = (float *) calloc(contour1_num_spline,sizeof(float));

 d2_x = (float *) calloc(contour2_num_spline,sizeof(float));
 d2_y = (float *) calloc(contour2_num_spline,sizeof(float));
 d2_z = (float *) calloc(contour2_num_spline,sizeof(float));
 dd2_x = (float *) calloc(contour2_num_spline,sizeof(float));
 dd2_y = (float *) calloc(contour2_num_spline,sizeof(float));
 dd2_z = (float *) calloc(contour2_num_spline,sizeof(float));
 curvature2 = (float *) calloc(contour2_num_spline,sizeof(float));

/* if(argc != 4)
  {
   fprintf(stdout, "\n** You forgot to specify file names **\n");
```

```
    fprintf(stderr, "\n%s%s%s%s%s\n\n%s\n\n",
            "Usage:  ", argv[0], " input_file1 ", "inout_file2", " output_file ",
            "(Ex: resample [contour1] [contour2] [out_contour])" );
    exit(0);
  }         */

/* fp1 = gfopen(argv[1], "r+");
 fp2 = gfopen(argv[2], "r+");
 outfp = gfopen(argv[3], "w");      */

/* Testing cases of the spline fitted data with specified names */

 fp1 = gfopen("sort1", "r+");
 fp2 = gfopen("sort2", "r+");
 outfp1 = gfopen("spline1.dat", "w");
 outfp2 = gfopen("spline2.dat", "w");

 path1_fp = gfopen("path1.dat", "w");
 path2_fp = gfopen("path2.dat", "w");

 /* Scan the contour file to see how many data points are there */

 while(fscanf(fp1, "%f%f%f", &tempx, &tempy, &tempz) != EOF)
        contour1_num++;

 while(fscanf(fp2, "%f%f%f", &tempx, &tempy, &tempz) != EOF)
        contour2_num++;

        printf("contour_1=%d contour_2=%d\n", contour1_num, contour2_num);

 rewind(fp1); /* Rewind the data files */
 rewind(fp2);

 /* Allocate the memory size of the contour points */
        contour1_pts_x = (float *) calloc(contour1_num,sizeof(float));
        contour1_pts_y = (float *) calloc(contour1_num,sizeof(float));
        contour1_pts_z = (float *) calloc(contour1_num,sizeof(float));
        contour2_pts_x = (float *) calloc(contour2_num,sizeof(float));
        contour2_pts_y = (float *) calloc(contour2_num,sizeof(float));
        contour2_pts_z = (float *) calloc(contour2_num,sizeof(float));

/* Scan the two sorted input files and perform the fitting process */

  for (i=0; i<contour1_num; i++) {
        fscanf(fp1, "%f %f %f", &temp1, &temp2, &temp3);

         contour1_pts_x[i] = temp1;
         contour1_pts_y[i] = temp2;
         contour1_pts_z[i] = temp3;

        printf("%f %f %f\n", contour1_pts_x[i],
               contour1_pts_y[i], contour1_pts_z[i]);
}

for (i=0; i<contour2_num; i++) {
        fscanf(fp2, "%f %f %f", &temp1, &temp2, &temp3);
         contour2_pts_x[i] = temp1;
         contour2_pts_y[i] = temp2;
         contour2_pts_z[i] = temp3;

        printf("%f %f %f\n", contour2_pts_x[i],
               contour2_pts_y[i], contour2_pts_z[i]);
}
       /* Definition of the path length array */
        path_length_1 = (float *) calloc(contour1_num , sizeof(float));
        path_length_2 = (float *) calloc(contour2_num , sizeof(float));

        path_length_1[0] = 0;


/* Store the path length information of two curves into buffers */

for (i=1; i<contour1_num; i++) {
        path_length_1[i] = sqrt(SQ(contour1_pts_x[i]-contour1_pts_x[i-1])
                                +SQ(contour1_pts_y[i]-contour1_pts_y[i-1])
                                +SQ(contour1_pts_z[i]-contour1_pts_z[i-1]))
                           +path_length_1[i-1];
}

for (i=0; i<contour1_num; i++)
        fprintf(path1_fp, "path1 = %f\n", path_length_1[i]);

        path_length_2[0] = 0;

for (i=1; i<contour2_num; i++) {
        path_length_2[i] = sqrt(SQ(contour2_pts_x[i]-contour2_pts_x[i-1])
                                +SQ(contour2_pts_y[i]-contour2_pts_y[i-1])
                                +SQ(contour2_pts_z[i]-contour2_pts_z[i-1]))
```

```
                        +path_length_2[i-1];
}

for (i=0; i<contour2_num; i++)
        fprintf(path2_fp, "path2 = %f\n", path_length_2[i]);


for(i=0; i<contour1_num_spline; i++)
        p a t h _ l e n g t h _ 1 _ s p l i n e [ i ]    =
(path_length_1[contour1_num-1]/(contour1_num_spline-1)) * i;

for(i=0; i<contour2_num_spline; i++)
        p a t h _ l e n g t h _ 2 _ s p l i n e [ i ]    =
(path_length_2[contour2_num-1]/(contour2_num_spline-1)) * i;


/* Spline fit the two curves, x - AP , y - LAT, z - VERT */
/* This part is to deal with the spline fitting of the x, y, and z
   directions of contour 1 and contour 2 */

/* Contour 1 spline fitting procedure */

ispline_gen(path_length_1, contour1_pts_x, contour1_num,
            path_length_1_spline, contour1_spline_x, contour1_num_spline,
            d1_x, dd1_x);

ispline_gen(path_length_1, contour1_pts_y, contour1_num,
            path_length_1_spline, contour1_spline_y, contour1_num_spline,
            d1_y, dd1_y);

ispline_gen(path_length_1, contour1_pts_z, contour1_num,
            path_length_1_spline, contour1_spline_z, contour1_num_spline,
            d1_z, dd1_z);

/* Calculate the curvature of the spline fitted curves */

        for(i=0;i<contour1_num_spline; i++) {
                K1 = d1_y[i]*dd1_z[i] - dd1_y[i]*d1_z[i];
                K2 = d1_z[i]*dd1_x[i] - dd1_z[i]*d1_x[i];
                K3 = d1_x[i]*dd1_y[i] - dd1_x[i]*d1_y[i];
                K = sqrt(SQ(K1)+SQ(K2)+SQ(K3));
                S = pow(sqrt(SQ(d1_x[i])+SQ(d1_y[i])+SQ(d1_z[i])),3);
                printf("K=%f, S=%f\n",K,S);
                curvature1[i] = K/S;
        }

        for(i=0;i<contour1_num_spline; i++)

                fprintf(outfp1, "%f %f %f\n", contour1_spline_x[i],
                        contour1_spline_y[i], contour1_spline_z[i]); /* print out the spline
fitted coordinates */

 /* Contour 2 spline fitting procedure */
 ispline_gen(path_length_2, contour2_pts_x, contour2_num,
          path_length_2_spline, contour2_spline_x, contour2_num_spline,
          d2_x, dd2_x);

 ispline_gen(path_length_2, contour2_pts_y, contour2_num,
          path_length_2_spline, contour2_spline_y, contour2_num_spline,
          d2_y, dd2_y);

 ispline_gen(path_length_2, contour2_pts_z, contour2_num,
          path_length_2_spline, contour2_spline_z, contour2_num_spline,
          d2_z, dd2_z);

        for(i=0;i<contour2_num_spline; i++) {
                K1 = d2_y[i]*dd2_z[i] - dd2_y[i]*d2_z[i];
                K2 = d2_z[i]*dd2_x[i] - dd2_z[i]*d2_x[i];
                K3 = d2_x[i]*dd2_y[i] - dd2_x[i]*d2_y[i];
                K = sqrt(SQ(K1)+SQ(K2)+SQ(K3));
                S = pow(sqrt(SQ(d2_x[i])+SQ(d2_y[i])+SQ(d2_z[i])),3);
                curvature2[i] = K/S;
        }

        for(i=0;i<contour2_num_spline; i++)

                fprintf(outfp2, "%f %f %f\n", contour2_spline_x[i],
                        contour2_spline_y[i], contour2_spline_z[i]); /* print out the spline
fitted coordinates */

/* Comment out this part temperatorily because the same starting and ending points of
the curves will
  be selected

 for(i=0; i<contour1_num_spline-2;i++) {
        min = i;
        for(j=i+1; j<contour1_num_spline-1;j++)
                if(CURVE_1[j] < CURVE_1[min]) min = j;
        if(i==0) index_1 = min;
        swap(CURVE_1,min,i);
}
for(i=0; i<contour2_num_spline-2;i++) {
        min = i;
        for(j=i+1; j<contour2_num_spline-1;j++)
                if(CURVE_2[j] < CURVE_2[min]) min = j;
       if(i==0) index_2 = min;
        swap(CURVE_2,min,i);
}
printf("min1=%f min2=%f", CURVE_1[0], CURVE_2[0]);

x_difference = contour1_spline_x[index_1] - contour2_spline_x[index_2];
printf("\ndiff=%5.2f\n", x_difference);

for(i=0;i<contour2_num_spline-1;i++)
        contour2_spline_x[i] += x_difference;                */

 printf("This is the end of program !"); /* Terminate the program */

free(contour1_pts_x);
free(contour1_pts_y);
free(contour1_pts_z);
free(contour2_pts_x);
free(contour2_pts_y);
free(contour2_pts_z);

free(contour1_spline_x);
free(contour1_spline_y);
free(contour1_spline_z);
free(contour2_spline_x);
free(contour2_spline_y);
free(contour2_spline_z);

free(path_length_1_spline);
free(path_length_2_spline);

free(d1_x);
free(d1_y);
free(d1_z);
free(dd1_x);
free(dd1_y);
free(dd1_z);
free(curvature1);
```

```
 free(curvature2);

 fclose(fp1);
 fclose(fp2);
 fclose(outfp1);
 fclose(outfp2);
}

/* Interpolating cubic spline function for irregularly-spaced points
   input : Y1 is the list of irregular data points(len1 entries)
           Their path length coordinates are specified in X1
   Output : Y2 <- cubic spline sampled according to new path length X2(len2 entries)
            Assume that X1,X2 entries are monotonically increasing */

void ispline_gen(float *X1, float *Y1,int len1, float *X2, float *Y2, int len2,
                 float *dyofdx, float *d2yofdx2)
{
        int i,j;
        float *YD,A0,A1,A2,A3,x,dx,dy,p1,p2,p3;

        /* Compute 1st derivatives at each point -> YD */
        YD = (float *) calloc(len1, sizeof(float));
        getYD_gen(X1,Y1,YD,len1);

 /*     for(i=0; i<len1; i++)
                printf("YD[%d]=%f", i, YD[i]);

        dyofdx = (float *) calloc(len2, sizeof(float));
        d2yofdx2 = (float *) calloc(len2, sizeof(float));   */

/* the following segment is to calculate the path length of each curve */

        /* error checking */
        if(X2[0]<X1[0] || X2[len2-1]>X1[len1-1]){
                fprintf(stderr,"ispline_gen:Out of range");
                exit(0);
        }
        /*
        p1 is left endpoint of interval
        p2 is the resampling position
        p3 is right endpoint of interval
        j is input index of current interval
        */
        p3 = X2[0]-1; /* force coefficient initiation */
        for(i=j=0;i<len2;i++) {
                /* check the new interval */
                p2 = X2[i];
                if(p2>p3) {
                        /* find the interval which contains p2 */
                        for(;j<len1 && p2>X1[j]; j++);
                        if(p2<X1[j]) j--;
                        p1 = X1[j];
                        p3 = X1[j+1];

                        /* compute spline coefficients */
                        dx = 1.0/(X1[j+1]-X1[j]);
                        dy = (Y1[j+1]-Y1[j])*dx;
                        A0 = Y1[j];
                        A1 = YD[j];
                        A2 = dx*(3.0*dy-2.0*YD[j]-YD[j+1]);
                        A3 = dx*dx*(-2.0*dy+YD[j]+YD[j+1]);
                }
                /* Use Honer's rule to calculate cubic polymonial */
                x = p2 -p1;
                Y2[i] = ((A3*x+A2)*x+A1)*x+A0;
                /* Compute the 1st and 2nd derivatives based on cubic equation */
                dyofdx[i] = (3*A3*x+2*A2)*x+A1;
                d2yofdx2[i] = 6*A3*x+2*A2;

                printf("dyofdx=%f,d2yofdx2=%f\n",dyofdx[i],d2yofdx2[i]);/* Y2 is the
spline fit value */
        }

        free(YD);
        free(dyofdx);
        free(d2yofdx2);

}

/* YD <- Computed 1st derivative of data in X,Y(len entries)
   The not-a-knot boundary condition is used */

void getYD_gen(float *X, float *Y, float *YD, int len)
{
        int i;
        float h0,h1,r0,r1,*A,*B,*C;

        /* allocate memory for tridiagonal bands A,B,C */
        A = (float *) calloc(len, sizeof(float));
        B = (float *) calloc(len, sizeof(float));
        C = (float *) calloc(len, sizeof(float));

        /* init first row data */
        h0 = X[1]-X[0];                 h1 = X[2]-X[1];
        r0 = (Y[1]-Y[0])/h0;            r1 = (Y[2]-Y[1])/h1;
        B[0] = h1*(h0+h1);
        C[0] = (h0+h1)*(h0+h1);
        YD[0] = r0*(3*h0*h1+2*h1*h1)+r1*h0*h0;

        /*init tridiagonal bands A,B,C, and column vectorYD */
        /* YD wil later be used to return the derivatives */
        for(i=1; i<len-1;i++) {
                h0 = X[i]-X[i-1];               h1 = X[i+1]-X[i];
                r0 = (Y[i]-Y[i-1])/h0;                  r1 = (Y[i+1]-Y[i])/h1;
                A[i] = h1;
                B[i] = 2*(h0+h1);
                C[i] = h0;
                YD[i] = 3*(r0*h1+r1*h0);
        }
        /* last row */
        A[i] = (h0+h1)*(h0+h1);
        B[i] = h0*(h0+h1);
        YD[i] = r0*h1*h1+r1*(3*h0*h1+2*h0*h0);

        /* solve for the tridiagonal matrix:YD=YD*inv(tridiag matrix) */

        tridiag_gen(A,B,C,YD,len);
        free(A);
        free(B);
        free(C);
}

/* Gauss elimination with backsubstitution for general
   tridiagonal matrix with bands A,B,C and column vector D */

void tridiag_gen(float *A, float *B, float *C, float *D, int len)
{
        int i;
        float b,*F;

        F = (float *) calloc(len,sizeof(float));
        /* Gauss elimination;forward substitution */
        b = B[0];
        D[0] = D[0]/b;
        for(i=1; i<len; i++) {
                F[i] = C[i-1]/b;
                b = B[i] - A[i]*F[i];
                if(b==0) {
                        fprintf(stderr,"getYD_gen:divide by zero");
                        exit(1);
                }
                D[i] = (D[i]-D[i-1]*A[i])/b;
        }

        /* backsubstitution */
        for(i=len-2;i>=0;i--) D[i] -= (D[i+1]*F[i+1]);

        free(F);
}

/* Input and output files definition */

FILE *gfopen(char *fn, char *mode)
{
        FILE *fp;

        if ((fp = fopen(fn, mode)) == NULL) {
                fprintf(stderr, "Cannot open %s - bye !\n", fn);
                exit(1);
        }
        return(fp);
}

void swap(float a[],int i, int j)
{
        float temp;

        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
}
```

```
/*      WEIGHT.C    */
/* This program is to combine the contour data set and the anatomical points
   both in the CT/MR images and the digitized data.  Linear weighting method
   is used because of the importance of the anatomical structures and the
   minimization of contour line errors.  A final data file is formed which is
   the proper input format for the SVD or optimization algorithms
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define N 400 /* Define the number of the formatted data file
                 which will be used in the reorganization procedures
                 of new data file for the least-squared best fit
                 optimization programs */

/* Global variables */
int   numpoints_1=0, numpoints_2=0, numpoints_3=0, numpoints_4=0;
float points_1[N][3], points_2[N][3], points_3[N][3], points_4[N][3];
float buffer_1[N][3], buffer_2[N][3];

/* user prototypes */
FILE *gfopen(char *fn, char *mode);

/* MAIN PROGRAM STARTS HERE */

void main(int argc, char *argv[])
{
 int i, j, weight;
 FILE *fp1, *fp2, *fp3, *fp4, *outfp;
 float tempx, tempy, tempz;


/* if(argc != 4)
  {
   fprintf(stdout, "\n** You forgot to specify file names **\n");
   fprintf(stderr, "\n%s%s%s%s\n\n%s\n\n",
           "Usage:  ", argv[0], " input_files ", " output_file ",
           "(Ex: weight [input.dat_1] [input.dat_2] [output.dat])" );
   exit(1);
  }

 fp1 = gfopen(argv[1], "r+");
 fp2 = gfopen(argv[2], "r+");
```

```
outfp = gfopen(argv[3], "w");      */

/* Temperatary definition of the data files required for the creation
   of new data files for the fitting calculation */

fp1 = gfopen("ct2", "r+");
fp2 = gfopen("spline1.dat", "r+");
fp3 = gfopen("digi2", "r+");
fp4 = gfopen("spline2.dat", "r+");
outfp = gfopen("gygy", "w");

/* Scan the contour file to see how many data points are there */
while(fscanf(fp1, "%f%f%f", &tempx, &tempy, &tempz) != EOF)
        numpoints_1++;

while(fscanf(fp2, "%f%f%f", &tempx, &tempy, &tempz) != EOF)
        numpoints_2++;

printf("numpoints_1=%d numpoints_2=%d\n", numpoints_1, numpoints_2);

while(fscanf(fp3, "%f%f%f", &tempx, &tempy, &tempz) != EOF)
        numpoints_3++;

while(fscanf(fp4, "%f%f%f", &tempx, &tempy, &tempz) != EOF)
        numpoints_4++;

printf("numpoints_3=%d numpoints_4=%d\n", numpoints_3, numpoints_4);

rewind(fp1); /* Rewind the data files for inputs */
rewind(fp2);
rewind(fp3);
rewind(fp4);

/* Start to read those points in the data files */
for (i = 0; i < numpoints_1; i++) {
        fscanf(fp1, "%f %f %f", &points_1[i][0],
                &points_1[i][1], &points_1[i][2]);
}

for (i = 0; i < numpoints_2; i++) {
        fscanf(fp2, "%f %f %f", &points_2[i][0],
                &points_2[i][1], &points_2[i][2]);
}
```

```
for (i = 0; i < numpoints_3; i++) {
        fscanf(fp3, "%f %f %f", &points_3[i][0],
              &points_3[i][1], &points_3[i][2]);
}

for (i = 0; i < numpoints_4; i++) {
        fscanf(fp4, "%f %f %f", &points_4[i][0],
              &points_4[i][1], &points_4[i][2]);
}

        weight = (int) (numpoints_2/numpoints_1); /* Weighting factor for points , this
is based on the linear weighting method */

        printf("weight = %d, weight = %d\n", weight, weight);

/* Calculate and reformat the data for first data set */
        for (j = 0; j < numpoints_1; j++) {

                for (i = j*weight; i < (j+1)*weight; i++) {

                buffer_1[i][0] = points_1[j][0];
                buffer_1[i][1] = points_1[j][1];
                buffer_1[i][2] = points_1[j][2];
                }
        }

        for (i = 0; i < numpoints_2; i++) {
                buffer_1[i+numpoints_1*weight][0] = points_2[i][0];
                buffer_1[i+numpoints_1*weight][1] = points_2[i][1];
                buffer_1[i+numpoints_1*weight][2] = points_2[i][2];
        }

/* Calculate and reformat the data for first data set */
        for (j = 0; j < numpoints_3; j++) {

                for (i = j*weight; i < (j+1)*weight; i++) {

                buffer_2[i][0] = points_3[j][0];
                buffer_2[i][1] = points_3[j][1];
                buffer_2[i][2] = points_3[j][2];
                }
        }

        for (i = 0; i < numpoints_4; i++) {
                buffer_2[i+numpoints_3*weight][0] = points_4[i][0];
                buffer_2[i+numpoints_3*weight][1] = points_4[i][1];
                buffer_2[i+numpoints_3*weight][2] = points_4[i][2];
        }

                fprintf(outfp, "%d\n", numpoints_1*weight + numpoints_2);

        for (i = 0; i < numpoints_1*weight + numpoints_2; i++) {
                fprintf(outfp, "%f %f %f %f %f %f\n",
                        buffer_1[i][0], buffer_1[i][1], buffer_1[i][2],
                        buffer_2[i][0], buffer_2[i][1], buffer_2[i][2]);
        }
}

/* Input and output files definition, grace function */
FILE *gfopen(char *fn, char *mode)
{
        FILE *fp;

        if ((fp = fopen(fn, mode)) == NULL) {
                fprintf(stderr, "Cannot open %s - bye !\n", fn);
                exit(1);
        }
        return(fp);
}
```

REFERENCES

Ada90 Adams, L., Krybus, W., Meyer-Ebrecht, D., Rueger, R., Gilsbach, J.M., Moesges, R., Schloendorff, G., "Computer-assisted surgery," IEEE Computer Graphics and Applications, Vol. 10, No. 3, pp. 43-51, 1990

Aru87 Arun, K.S., Huang, T.S., Blostein, S.D., "Least-squares fitting of two 3-D point sets," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. 5, pp. 698-700, 1987

Att86 Attix, F.H., Introduction to radiological physics and radiation dosimetry, John Wiley & Sons, Inc., New York, 1986

Ayr90 Ayres F. Jr., Mendelson E., Theory and problems of differential and integral calculus 3/ed, Schaum's outline series. McGraw-Hill, Inc., New York, 1990

Bar93 Barnett, G.H., Kormos, D.W., Steiner, C.P., Weisenberger, J., "Use of a frameless, armless stereotactic wand for brain tumor localization with two-dimensional and three-dimensional neuroimaging," Neurosurgery, Vol. 33, No.4, pp. 674-678, 1993

Bet84 Betti, O., Derechinsky, V., "Hyperselective encephalic irradiation with linear accelerator," Acta Neurochirurgica, Suppl., Vol. 33, pp. 385-390, 1984

Boh86 Bohachevsky, I.O., Johnson, M.E., Stein, M.L., "Generalized simulated annealing for function optimization," Technometrics, Vol. 28, No. 3, pp. 209-217, 1986

Bov90 Bova, F.J., "Radiation Physics," Neurosurgery Clinics of North America, Vol. 1, No.4, pp. 909-931, 1990

Bov91 Bova, F.J., Friedman, W.A., "Stereotactic angiography: An inadequate database for radiosurgery ?," Int. J. Radiation Oncology Biol. Phys., Vol. 20, pp. 891-895, 1991

Bov93 Bova, F.J., Friedman, W.A., Mendenhall, W.M., "Stereotactic radiosurgery," Medical Progress through Technology, Vol. 18, pp. 239-251, 1993

Bro86 Brown, L.B., Merry, J.B., Wells, D.N., "Coordinate measurement with a tracking laser interferometer," Lasers and Applications, Oct., pp. 69-71, 1986

Byh78 Byhardt, R.W., Cox, J.D., Hornburg, A., Liermann, G., "Weekly localization films and detection of field placement errors," Int. J. Radiation Oncology Biol. Phys., Vol. 4, pp. 881-887, 1978

Col85 Colombo, F., Benedetti, A., Pozza, F., Zanardo, A., Avanzo, R.C., Chierego, G., Marchetti, C., "Stereotactic radiosurgery utilizing a linear accelerator," Appl. Neurophysiol., Vol.48, pp. 133-145, 1985

DeB78 De Boor, C., A practical guide to splines, Springer-Verlag, New York, 1978

Del91 Delannes, M., Daly, N.J., Bonnet, J., Sabatier, J., Tremoulet, M., "Fractionated radiotherapy of small inoperable lesions of the brain using a non-invasive stereotactic frame," Int. J. Radiation Oncology Biol. Phys., Vol. 21, pp. 749-755, 1991

Dub93 Dubois, D., "Digitization of film for quality assurance and dosimetry," Master's thesis, University of Florida, Gainesville, 1993

Fit75 Fitzgerald, L.T., Mauderli, W., "Analysis of errors in three-dimensional reconstruction of radium implants from stereo radiographs," Radiology, Vol. 115, pp. 455-458, 1975

Fri89a Friedman, W.A., Bova, F.J., "The University of Florida radiosurgery system," Surg. Neurol., Vol. 32, pp. 334-342, 1989

Fri90 Friedman, W.A., "LINAC radiosurgery," Neurosurg Clin North Am 1:991, 1990

Fri92a Friedman, W.A., Bova, F.J., "Linear accelerator radiosurgery for arteriovenous malformations," J. Neurosurgery Vol. 77, pp. 832-841, 1992

Fri92b Friedman, W.A., Bova, F.J., Spiegelmann, R., "Linear accelerator radiosurgery at the University of Florida," Neurosurg Clinics of North America., Vol. 3, pp. 141-166, 1992

Fri89b Friets, E.M., Strohbehn, J.W., Hatch, J.F., Roberts, D.W., "A frameless stereotaxic operating microscope for neurosurgery," IEEE Trans. on Biomed. Imag., Vol. 36, No.6, pp. 608-617, 1989

Gei87 Geijn, J.V., "Modeling of the macro-response of tumors and surrounding normal tissues of fractionated radiation treatment," The use of computers in radiation treatment, Elsevier Science Publishers B. V. (North Holland), 1987

Ger82 Gerber, R.L., Marks, J.E., Purdy, J.A., "The use of thermal plastics for immobilization of patients during radiotherapy," Int. J. Radiation Oncology Biol. Phys., Vol. 8, pp. 1461-1462, 1982

Gil91 Gill, S.S., Thomas, D.G.T., Warrington, A.P., Brada, M., "Relocatable frame for stereotactic external beam radiotherapy," Int. J. Radiation Oncology Biol. Phys., Vol. 20, pp. 599-603, 1991

Goi83 Goitein, M., Abrams, M., "Multi-dimensional treatment planning: II. Beam's Eye-View, back projection, and projection through CT sections," Int. J. Radiation Oncology Biol. Phys., Vol. 9, pp. 789-797, 1983

Goi85 Goitein, M., "Calculation of the uncertainty in the dose delivered during radiation therapy," Med. Phys., Vol. 12, No. 5, pp. 608-612, 1985

Goi86 Goitein, M., "Causes and consequences of inhomogeneous dose distributions in radiation therapy," Int. J. Radiation Oncology Biol. Phys., Vol. 12, pp. 701-704, 1986

Gol89 Golub, G.H., Van Loan, C.F., Matrix multiplications, The Johns Hopkins University Press, Baltimore, 1989

Gon87 Gonzales, R.C., Wintz, P., Digital image processing, 2nd ed., Addison-Wesley, Reading, Mass., 1987

Gra91 Graham, J.D., Warrington, A.P., Gill, S.S., Brada, M., "A non-invasive, relocatable stereotactic frame for fractionated radiotherapy and multiple imaging," Radiotherapy and Oncology, Vol. 21, pp. 60-62, 1991

Hal93 Hall, E. J., Brenner, D. J., "The radiobiology of radiosurgery: rationale for different treatment regimes for AVMs and malignancies," Int. J. radiation Oncology Biol. Phys., Vol. 25, pp. 381-385, 1993

Har90 Hariz, M.I., Henriksson, R., Lofroth, P-O., Laitinen, L.V., Saterborg, N-E., "A non-invasive method for fractionated stereotactic irradiation of brain tumors with linear accelerator," Radiotherapy and Oncology, Vol. 17, pp. 57-72, 1990

Har85 Hartmann, G.H., Schlegel, W., Sturm V., Kober, B., Pastyr, O., Lorenz, W.J., "Cerebral radiation surgery using moving field irradiation at a linear accelerator facility," Int. J. Radiation Oncology Biol. Phys., Vol. 11, pp. 1185-1192, 1985

Hau72 Haus, A.G., Marks, J.E., "Detection and evaluation of localization errors in patient radiation therapy," Invest. Radiol., Vol. 8, pp. 834, 1972

Hei89 Heifetz, M.D., Whiting, J., Bernstein, H., Wexler, M., Rosemark, P., Thompson, R.W., "Stereotactic radiosurgery for fractionated radiation: a proposal applicable to linear accelerator and proton beam programs," Stereotactic Funct. Neurosurg Vol. 53, pp. 167-177, 1989

Hen82 Hendrickson, F.R., "Precision in radiation therapy," Int. J. Radiation Oncology Biol. Phys., Vol. 8, pp. 311-312, 1982

Hol91 Holleb, A.I., Fink, D.J., Murphy, G.P., Clinical oncology, American Cancer Society, Atlanta, GA, 1991

Hou85 Houdek, P.V., Fayos, J.V., VanBuren, J.M., Ginsberg, M.S., "Stereotaxic radiotherapy technique for small intracranial lesions," Medical Physics, Vol. 12, pp. 469-472, 1985

Hou91 Houdek, P.V., Schwade, J.G., Serago, C.F., Landy, H.J., Pisciotta, V., Wu, X., Markoe, A.M., Lewin, A.A., Abitbol, A.A., Bujnoski, J.L., Marienberg, E.S., Fiedler, J.A., Ginsberg, M.S., "Computer controlled stereotaxic radiotherapy system," Int. J. Radiation Oncology Biol. Phys., Vol. 22, pp. 175-180, 1991

Hua86 Huang, T.S., Blostein, S.D., Margerum, E.A., "Least-squares estimation of motion parameters from 3-D point correspondences," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 2, pp. 198-201, 1986 198-201, 1986

Hui87 Huizenga, H., Levendag, P.C., De Porre, P.M.Z.R., Visser, A.G., "Accuracy in radiation field alignment in head and neck cancer: A prospective study," Radiotherapy and Oncology, Vol. 11, pp. 181-187, 1987

Hul89 Hulshof, M., Vanuytsel, L., Bogaert, W.V.D., Schueren, E.V.D., "Localization errors in mantle-field irradiation for Hodgkin's Disease," Int. J. Radiation Oncology Biol. Phys., Vol. 17, pp. 679-683, 1989

Jon90 Jones, D., Christopherson, D.A., Washington, J.T., Hafermann, M.D., Rieke, J.W., Travaglini, J.J., Vermeulen, S.S., "A technique for fractionated external beam stereotactic radiotherapy," submitted to Int. J. Radiation of Radiation Biol. Phys., 1990

Kat91 Kato, A., Yoshimine, T., Hayakawa, T., Tomita, Y., Ikeda, T., Mitomo, M., Harada, K., Mogami, H., "A frameless, armless navigational system for computer-assisted neurosurgery," J. Neurosurg. Vol. 74, pp. 845-849, 1991

Ken89 Kendall, E.A., An introduction to numerical analysis, John Wiley & Sons, New

York, 1989

Kes91 Kessler, M.L., Pitluck, S., Petti, P., Castro, J.R., "Integration of multimodality imaging data for radiotherapy treatment planning," Int. J. Radiation Oncology Biol. Phys., Vol. 21, pp. 1653-1667, 1991

Kha84 Khan, F.M., The physics of radiation therapy, Williams and Wilkins, New York, 1984

Kir83 Kirkpatrick, S., Gelatt, C.D., Jr., Vecchi, M.P., "Optimization by simulated annealing," Vol. 220, No. 4598, pp. 671-680, 1983

Kje77 Kjellberg, R.N., Poletti, C.E., Roberson, G.H., and Adams, R.D., "Bragg-peak proton beam treatment for arteriovenous malformation of the brain," Exc. Med., Vol. 433, pp. 181-186, 1977

Lam86 Lam, K.S., Partowmah, M., Lam, W.C., "An on-line electronic portal imaging system for external beam radiotherapy," Br. J. Radiol., Vol. 59, pp. 1007-1013, 1986

Lau85 Lau, K., Hocken, R., Haynes, L., "Robot performance measurements using automatic laser tracking techniques," Robotics and Computer-Integrated Manufacturing, Vol. 2, No.3, pp. 227-236, 1985

Lek51 Leksell, L., "The stereotaxic method and radiosurgery of the brain," Acta Chir. Scand., Vol. 102, pp. 316-319, 1951

Lek71 Leksell, L., Stereotaxis and Radiosurgery, Charles C. Thomas, Springfield, Ill., 1971

Lin86 Lin, Z.C., Huang, T.S., Blostein, S.D., Lee, H., Margerum, E.A., "Motion estimation from 3-D point sets with and without correspondences," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 2, pp. 194-198, 1986

Liu88 Liu, Y., Huang, T.H., "A linear algorithm for motion estimation using straight line correspondences," Computer Vision, Graphics, and Image Processing. Vol. 44, pp. 35-37, 1988

Liu90 Liu, Y., Huang, T.H., Faugeras, O.D., "Determination of camera location from 2-D to 3-D line and point correspondences," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 1, pp. 28-37, 1990

Loe90 Loeffler, J.S., Siddon, R.L., Wen, P.Y., Nedzi, L.A., Alexander, E., "Stereotactic radiosurgery of the brain using a standard linear accelerator: a study of early and late effects," Radiotherapy and Oncology, Vol. 17, pp. 311-321, 1990

Lut88 Lutz, W., Winston, K.R., Maleki, N., "A system for stereotactic radiosurgery with a linear accelerator," Int. J. Radiation Oncology Biol. Phys., Vol. 14, pp. 373-381, 1988

Lym89 Lyman, J.T., Philips, M.H., Frankel, K.A., Fabrikant, J.I., "Stereotactic frame for neuroradiology and charged particle Bragg peak radiosurgery of intracranial disorders," Int. J. Radiation Oncology Biol. Phys., Vol. 16, pp. 1615-1621, 1989

Mar74 Marks, J.E., Haus, A.G., Sutton, H.G., Griem, M.L., "Localization error in the radiotherapy of Hodgkin's Disease and malignant lymphoma with extended mantle fields," Cancer Vol. 34, pp. 83-90, 1974

Mar76 Marks, J.E., Haus, A.G., Sutton, H.G., Griem, M.L., "The value of frequent treatment verification films in reducing localization error in the radiation of complex fields," Cancer Vol. 37, pp. 2755-2761, 1976

Mee90 Meertens, H., Bijhold, J., Strackee, J., "A method for the measurement of field placement errors in digital portal images," Phys. Med. Biol., Vol. 35, pp. 299-323, 1990

Met53 Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., "Equation of state calculations by fast computing machines," The Journal of Chemical Physics, Vol. 21, No. 6, pp. 1087-1092, 1953

Moo91 Mooring, B.W., Roth, Z.S., Driels, M.R., Fundamentals of manipulator calibration, John Wiley & Sons, Inc.,New York, 1991

Mos79 Moss, W.T., Brand, W.N., Battifora, H.(eds), Radiation oncology : rationale, technique, results, 5th ed., pp. 24-25, St. Louis, CV Mosby, 1979

Nel65 Nelder, J.A., Mead, R., "A simplex method for function minimization," Computer Journal, Vol. 7, p 308, 1965

Nor93 Northern Digital, Inc., OPTOTRAK/3000 series Motion Measurement System, Waterloo, Ontario, Canada, 1993

Pel87 Pelizzari, C.A., Chen, G.T.Y., "Registration of multiple diagnostic imaging scans using surface fitting," The Use of Computers in Radiation Treatment, Elsevier Science Publishers B. V. (North Holland), 1987

Pel89 Pelizzari, C.A., Chen, G.T.Y., Spelbring, D.R., Weichselbaum, R.R., Chen, C.T., "Accurate three-dimensional registration of CT, PET, and/or MR images of the brain," J. of Computer Assisted Tomography, Vol. 13, pp. 20-26, 1989

Pel91 Pelizzari, C.A., Tan, K.K., Levin, D.N., Chen, G.T.Y., Balter, J., "Interactive 3D patient-image registration," Proceedings, Information Processing in Medical Imaging, 12th Inter. Conf., IPMI'91, Wye, UK, 1991

Per87 Perez, C.A., Brady, L.W., "Principles and practice of radiation oncology," pp.3-11, J. B. Lippincott Co., 1987

Pod89 Podgorsak, E.B., Pike, G.B., Olivier, A., Pla, M., and Souhami, L., "Radiosurgery with high energy photons beams: a comparison among techniques," Int. J. Radiation Oncology Biol. Phys. Vol. 16, pp. 857-865, 1989

Pre92 Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., Numerical recipes: The art of scientific computing, Cambridge University Press, Cambridge, 1992

Pro91 Product Genesis Inc. (PGI), patient Positioning System Description, 1991

Pur90 Purdy, J., Gerber, R., Harms, W., "Patient positioning, immobilizing and treatment verification," (unpublished paper), 1990

Rab85 Rabinowitz, I., Broomberg, J., Goitein, M., McCarthy, K., Leong, J., "Accuracy of radiation field alignment in clinical practice," Int. J. Radiation Oncology Biol. Phys. Vol. 11, pp. 1857-1867, 1985

Rei87 Reinstein, L.E., Meek, A.G., "Workshop on geometric accuracy and reproducibility in radiation therapy," Int. J. Radiation Oncology Biol. Phys. Vol. 13, pp. 809-810, 1987

Rog76 Rogers D.F., Adams J.A., Mathematical elements for computer graphics, McGraw-Hill, Inc., 1976

Ros91 Rosenman, J., Sailer, S.L., Sherouse, G.W., Chaney, E.L., Tepper, J.E., "Virtual simulation: initial clinical results," Int. J. Radiation Oncology Biol. Phys. Vol. 20, pp. 843-851, 1991

She90 Sherouse, G.W., Bourland, J.D., Reynolds, K., McMurry, H.L., Mitchell, T.P., Chaney, E.L., "Virtual simulation in the clinical setting: some practical considerations," Int. J. Radiation Oncology Biol. Phys. Vol. 19, pp. 1059-1065, 1990

Sho87 Shoup, T.E., Mistree, F., Optimization methods-with applications for personal computers, Prentice-Hall, Inc., 1987

Sid87 Siddon, R.L., Barth, N.H., "Stereotaxic localization of intracranial targets," Int. J. Radiation Oncology Biol. Phys. Vol. 13, pp. 1241-1246, 1987

Spi91 Spiegelmann R, Friedman W.A., "The radiosurgical treatment of meningiomas." In Schmidek H (ed): Meningiomas and their surgical treatment, Baltimore, Williams & Wilkins, 1991

Stu87 Sturm, V., Kober, B., Hover, K-H., Schlegel, W., Boesecke, R., Pastyr, O., Hartmann, G.H., Schabbert, S., Winkel, K.Z., Kunze, S., Lorenz, W.J., "Stereotactic percutaneous single dose irradiation of brain metastases with a linear accelerator," Int. J. Radiation Oncology Biol. Phys. Vol. 13, pp. 279-282, 1987

Tan93 Tan, K.K., Grzeszczuk, R., Levin, D., Pelizzari, C.A., Chen, T.Y., Erickson, R.K., Johnson, D., Dohrmann, G.J., "A frameless stereotactic approach to neurosurgical planning based on retrospective patient-image registration," J Neurosurgery Vol. 79, pp. 296-303, 1993

Tas83 Task Group 21, Radiation Therapy Committee, American Association of Physicists in Medicine, "A protocol for the determination of absorbed dose from high-energy photon and electron beam," Med. Phys., Vol. 10, No. 6, pp. 741-771, 1983

Tho91a Thornton, A.F., Ten Haken, R.K., Gerhardsson, A., Correll, M., "Three-dimensional motion analysis of an improved head immobilization system for simulation, CT, MRI, and PET imaging," Radiotherapy and Oncology, Vol. 20, pp. 224-228, 1991

Tho91b Thornton, A.F., Jr., Ten Haken, R.K., Weeks, K.J., Gerhardsson, A., Correll, M., Lash, K.A., "A head immobilization system for radiation simulation, CT, MRI, and PET imaging," Med. Dosim. Vol. 16, pp. 51-56, 1991

Tob64 Tobias, C.A., Lawrence, J.H., Lyman, J.T., Born, J.L., Gottschalk, A., Linfoot, J. and McDonald, J. "Process report on pituitary irradiation." In Haley, T.J., Snider, R.S., (eds.) Response of the nervous system to ionizing radiation, Boston, Little, Brown and Company, pp. 19-35, 1964

Van84 Vanderplaats, G.N., Numerical optimization techniques for engineering design: with applications, McGraw-Hill Company, New York, 1984

Ver82 Verhey, L.J.,Goitein, M., McNulty, P., Munzenrider, J.E., Suit, H.D., "Precise positioning of patients for radiotherapy," Int. J. Radiation Oncology Biol. Phys.

Vol. 8, pp. 289-294, 1982

Wat87 Watanabe, E., Watanabe, T., Manaka, S., Mayanagi, Y., Takakura, K., "Three-dimensional digitizer (Neuronavigator): New equipment for computed tomography-guided stereotaxic surgery," Surg. Neurol. Vol. 27, pp. 543-547, 1987

Wil79 Williamson, T.J., "Improving the reproducibility of lateral therapy portal placement," Int. J. Radiation Oncology Biol. Phys. Vol. 5, pp. 407-409, 1979

Wol92 Wolberg, G., Digital imaging warping, IEEE Computer Society Press, Los Alamitos, California, 1992

Yae90 Yaes, R.J., "The biological effect of inhomogeneous dose distributions in fractionated radiotherapy," Int. J. Radiation Oncology Biol. Phys. Vol. 19, pp. 203-207, 1990

Zhu92 Zhuang, H., Li, B., Merry., J.B., Wells, D.N., "Self-calibration and mirror center offset elimination of a multi-beam laser tracking system," submitted to Journal of Robotic and Autonomous Systems, special issue on "Trends in robot kinematics, dynamics, control, sensing and computer programming," 1992

BIOGRAPHICAL SKETCH

Ching-Chong J. Yang was born on November 24, 1962 in Taiwan, ROC. He graduated from Cheng-Kuo High School in 1982. He then went to Shin-Chu, a city 50 miles from Taipei, entered the National Tsing-Hua University, and graduated with a Bachelor of Science degree in nuclear engineering in 1986. After working for Pacific Engineers and Constructors Ltd. as a nuclear engineer in high level waste research, he joined China Technical Consultants Inc. as an environmental engineer responsible for the low level waste final disposal project. He came to the USA in 1988 for advanced study in the medical physics program at the University of Missouri-Columbia, resulting in a Master of Science degree in 1990. He then transferred to the University of Florida for the medical physics Ph.D. program, where he has been a clinical assistant and has been involved with the University of Florida Stereotactic Radiosurgery System.

He was a scholarship winner from China Technical Consultants, Inc. and project leader for low level waste final disposal in Taiwan. He is currently a student member of the American Association of Physicists in Medicine.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Frank J. Bova, Chair
Associate Professor of
Nuclear Engineering Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Lawrence T. Fitzgerald
Associate Professor of
Nuclear Engineering Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

James S. Tulenko
Professor of Nuclear Engineering Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

William S. Properzio
Associate Professor of
Nuclear Engineering Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

William A. Friedman
Professor of Neuroscience

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

William M. Mendenhall
Professor of Radiation Oncology

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Carl D. Crane
Associate Professor of
Mechanical Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

April 1994

[illegible]

[illegible] Winfred M. Phillips
Dean, College of Engineering

Karen A. Holbrook
Dean, Graduate School

LD
1780
1994
.Y22